Geautomatiseerde videoadaptatie
gebaseerd op tijdsafhankelijke contextparameters

Automated Video Adaptation
Based on Time-Varying Context Parameters

Robbie De Sutter

Promotor: prof. dr. ir. R. Van de Walle
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Faculteit Ingenieurswetenschappen
Academiejaar 2005 - 2006

UNIVERSITEIT
GENT

ISBN 90-8578-099-3
NUR 965
Wettelijk depot: D/2006/10.500/57

Dankwoord

Met het indienen van dit doctoraat sluit ik een periode af van bijna vijf jaar. Tijdens deze periode heb ik de kans gekregen om in een uitzonderlijk boeiend domein aan wetenschappelijk onderzoek te doen. Ik heb prachtige kansen en gelegenheden gekregen om met uiterst interessante mensen te mogen discussiëren en dit de wereld rond op vele internationale conferenties en meetings. Het bijwonen van dergelijke conferenties en meetings heeft me als onderzoeker en als mens enorm veel bijgebracht. Ik neem dan ook graag van de gelegenheid gebruik om de mensen te bedanken zonder wie dit allemaal niet mogelijk zou zijn geweest.

Eerst en vooral wens ik mijn promotor, prof. Rik Van de Walle, te bedanken. Het is dankzij hem dat het kleine Multimedia Lab waarin ik als onderzoeker ben begonnen in 2001 kon uitgroeien tot een internationaal gerespecteerde groep. Ik dank ook mijn promotor voor de vele feedback- en discussiemomenten in de loop van mijn onderzoek en tijdens het schrijven van deze thesis, zelfs op de momenten wanneer de omstandigheden niet altijd optimaal waren. Tenslotte dank ik mijn promotor ook om mij de mogelijkheid te hebben gegeven om me niet uitsluitend als onderzoeker te laten ontplooien.

Also, I would like to thank prof. Hermann Hellwagner (University of Klagenfurt – ITEC, Austria). He gave me the opportunity to conduct research at his lab for a period of three months in a very idyllic setting. At the same time, I wish to thank my former colleagues at ITEC for their help during my research, but also for their hospitality and their guided tours. Most of the work discussed in the fourth chapter was realized during this visit.

Daarnaast wil ik mijn collega's van Multimedia Lab en Medisip bedanken. Mede dankzij hun feedback, discussies en hulp is dit doctoraat

gevormd. Zonder hun kennis en de opbouwende kritiek was dit boek er nooit gekomen. Bovendien zorgden ze ervoor dat het nooit saai was op kantoor zodat ik me er altijd thuis heb gevoeld. Verder wil ik de collega's die delen uit dit boek hebben nagelezen, die één of meer van mijn papers hebben nagelezen of die talloze Excel vakjes hebben zitten inkleuren nog eens extra te bedanken voor de geleverde inspanningen.

Graag wil ik ook Rita Breems en Ellen Lammens langs deze weg bedanken. Dankzij hen is de administratieve papierberg aan mij grotendeels voorbijgegaan (oef). Maar ik wil hen zeker ook bedanken voor de vele gesprekken aan de koffiemachine of tijdens de lunchpauzes.

Dit dankwoord is ook een uitgelezen kans om mijn ouders en mijn broer te bedanken, vooral omdat zij altijd in mij zijn blijven geloven en blijven steunen ook al ging het eens wat minder vlot. Ik wil mijn ouders in het bijzonder danken voor de mogelijkheden en hulp die ze me gegeven hebben om me op mijn eigen manier verder te ontplooien.

Ook wil ik mijn vrienden van harte bedanken voor de vele steun (en afleiding) over de voorbije jaren. Ik dank hen dat ze me af en toe (letterlijk) uit mijn huis haalden om te ontspannen en te genieten van het leven. Nu het doctoraat afgerond is, hoop ik wat meer tijd voor hen te hebben.

Tot slot wil ik Jan bedanken, niet in het minst omdat hij alles zo grondig heeft nagelezen en voor zijn niet aflatende begrip en motivatie tijdens mijn doctoraatsonderzoek, ook tijdens de moeilijkere momenten. Jan, je bent voor mij een zeer belangrijke echte steun geweest het voorbije jaar. Dat er nog vele jaren mogen volgen!

Robbie De Sutter
14 september 2006

Samenvatting

Aangezien het Internet voortdurend uitbreidt met nieuwe multimediale data die afspeelbaar is op nieuwe types van toestellen en die verstuurd wordt over nieuwe soorten netwerken, is het noodzakelijk dat er actie ondernomen wordt zodat gebruikers deze data overal, altijd en op eender welke manier kunnen raadplegen. Dit gebeurt momenteel door adhocoplossingen. Neem bijvoorbeeld een Internet-gebaseerde *Video-op-Aanvraag* toepassing die audiovisuele data over IP-gebaseerde netwerken stroomt. Momenteel moet een eindgebruiker, die gebruik wil maken van deze toepassing, vooraf verschillende (technische) vragen beantwoorden over zijn toestel, zijn netwerkconnectie en zijn gebruikersvoorkeuren – met andere woorden vragen over zijn *gebruikersomgeving*. Daarna kan hij een versie kiezen die best geschikt is voor deze gebruikersomgeving. De aanbieder van de audiovisuele data moet per aangeboden video meerdere versies hebben die uitsluitend verschillen op technisch vlak, bijvoorbeeld verschillende resoluties, zodoende zoveel mogelijk verschillende gebruikersomgevingen te ondersteunen. Dit heeft als gevolg dat de eindgebruiker een versie ontvangt die niet geoptimaliseerd is voor zijn toestel en dat de aanbieder meerdere versies van de audiovisuele data moet onderhouden. Het spreekt voor zich dat deze oplossing onhoudbaar is. Inderdaad, aangezien meer en meer nieuwe toestellen en netwerktechnologieën beschikbaar komen met elk hun eigen karakteristieken, moet de videoaanbieder steeds meer versies ondersteunen per video of moet de gebruiker zich tevreden stellen met een versie die minder geschikt is voor zijn toestel.

Het *Universele Multimediatoegang* (Universal Multimedia Access, UMA) raamwerk probeert voor dit probleem een oplossing aan te bieden. Het doel is het mogelijk maken van de consumptie van multimediale data in verschillende gebruikersomgevingen door middel van het creëren van verschillende presentaties vanaf één bron.

Het doel van deze thesis is het onderzoeken welke de vereisten zijn om een UMA-compatibele architectuur en toepassing te realiseren, zoals de Video-op-Aanvraag toepassing, en om de verschillende problemen die hierbij ontstaan op te lossen.

Eerst wordt er bestudeerd hoe er op een gestandaardiseerde manier kan beschreven worden *wat* er geconsumeerd wordt – de *inhoud* – en *hoe* het geconsumeerd wordt – de *context*. Om het vergelijken van de verschillende internationale standaarden voor het beschrijven van de inhoud mogelijk te maken, wordt er een objectief evaluatie- en vergelijkingsmodel gedefinieerd. Dit model helpt bij het selecteren van de best geschikte standaard voor een welbepaalde toepassing. Voor de UMA-compatibele architectuur is dat de *MPEG-7* specificatie. Vervolgens worden er drie standaarden voor het beschrijven van de contextinformatie bestudeerd. Hieruit blijkt dat de *MPEG-21 Digitale Item Adaptatie* standaard de meest generieke en allesomvattende standaard is. Ondanks het feit dat deze standaard zo omvangrijk is, wordt er een applicatiesuite ontwikkeld die bruikbaar is op kleine, gelimiteerde toestellen (bijvoorbeeld een GSM) en waarmee het mogelijk is om MPEG-21 compatibele berichten te lezen, aan te passen en uit te schrijven.

De informatie over de inhoud en de context wordt gebruikt voor het aanpassen van de audiovisuele datastroom zodat de resulterende geoptimaliseerde versie bruikbaar is in de gegeven context. Deze thesis behandelt niet hoe een beslissing kan genomen worden over welke de gewenste aanpassing is, maar bestudeert wel de vereisten zodat deze beslissing kan genomen worden. Met andere woorden, er wordt nagegaan hoe de informatie over de inhoud en de context uitgewisseld kan worden met een component die een beslissing kan nemen. Dit wordt gerealiseerd door de negotiatie te bekijken als het aanroepen van een functie op afstand. Aansluitend wordt een belangrijke uitbreiding op UMA geïntroduceerd, namelijk het concept van *tijdsafhankelijke metadata*. Het basisidee is het dynamisch aanpassen van de multimedia data op basis van een veranderende context door het hernegotiëren van de contextinformatie. Op deze manier resulteren de aanpassingen in de gebruikersomgeving in het dynamisch heroptimaliseren van de multimediale data.

Ook het aanpassen van de audiovisuele stromen is een belangrijk onderdeel in deze thesis. Na een overzicht van de huidige stand van zaken met betrekking tot de schaalbare videotechnieken en -standaarden, wordt er in het bijzonder aandacht besteed aan regiogebaseerde videocodering. Bij deze techniek wordt een bepaald deel van de

videoscène als belangrijker aanzien dan de rest van het beeld. MPEG-4 FGS maakt het mogelijk om deze regio visueel te verbeteren. Deze standaard ondersteunt echter alleen een vaste zone voor de volledige videosequentie. Daarom worden er nieuwe en snelle algoritmen ontwikkeld die het mogelijk maken om bewegende objecten in een videosequentie automatisch te volgen. Deze algoritmen werken in het gecodeerde domein en maken gebruik van de bewegingsvectoren. Verschillende testen worden uitgevoerd om de kwaliteit na te gaan. Een complexiteitsanalyse bewijst bovendien dat de methoden voldoende snel en bruikbaar zijn in ware-tijdstoepassingen. Bovendien zijn de algoritmen generiek en bruikbaar in andere videocompressiestandaarden.

Deze thesis geeft ook een oplossing voor de belangrijkste problemen van XML-gebaseerde data, zoals de inhouds- en contextinformatie, namelijk de overheadkosten en het ontbreken van ondersteuning voor aanpassingen. Eerst wordt er onderzoek verricht naar de verschillende manieren om XML-gebaseerde data te verwerken. Dit resulteert in een globaal overzichtsmodel van XML ontleders en gebaseerd op deze studie wordt er een *notatie-onafhankelijke XML ontleder* ontwikkeld. Applicaties kunnen deze ontleder gebruiken om XML-gebaseerde data te verwerken zonder dat ze hoeven te weten welke notatie er werd gebruikt. Op deze manier is het eenvoudig om een compacte (binaire) XML notatie te gebruiken in plaats van de klassieke tekstuele notatie. Drie technieken (ZIP-compressie, ASN.1-PER en MPEG-B BiM) worden geëvalueerd als potentieel alternatief en dit op basis van hun compressie-efficiëntie (met andere woorden de reductie van de overheadkosten) en hun onmiddellijke inzetbaarheid in toepassingen. Uit het onderzoek blijkt dat BiM het beste scoort met betrekking tot de compressie-efficiëntie, maar een te hoge complexiteit heeft om onmiddellijk bruikbaar te zijn. Hoewel ZIP-compressie een lagere compressie-efficiëntie heeft, is deze techniek wel onmiddellijk bruikbaar als alternatieve XML notatie.

Tenslotte worden de besproken technieken geïntegreerd in een Video-op-Aanvraag applicatie die compatibel is met de UMA principes en die tijdsafhankelijke metadata ondersteunt. Hiermee hoop ik dat ik de lezer overtuigd heb dat de methoden en technieken besproken in deze thesis een initiële en originele bedrage hebben geleverd voor het ontwikkelen of het verbeteren van UMA-compatibele toepassingen.

Summary

As the Internet is continuously expanding with new content that is consumable on new devices and that can be transmitted over new types of networks, actions are needed to make transparent and ubiquitous content consumption possible anywhere, anytime, and anyhow. Currently, ad hoc solutions try to make this happen. Take, for example, an Internet-based *Video-on-Demand* application, i.e., the streaming of audio-visual content over IP-based networks. Today, if an end user wants to consume audio-visual content over the Internet, he first has to answer several technical questions about his device, the network connection, and his personal preferences – in short, questions about the consumption *context*. As a result, he can select a particular version of the video that more or less suits the context. The content provider has a *simulstore* containing a limited number of semantically equal videos with different technical characteristics, for example audio-visual content streams with different resolutions. The end user receives suboptimal content while the content provider has to maintain several similar content streams. This solution is inadequate and unsustainable in the long run, especially since more and more different kinds of end-user devices and network technologies become available to acquire and consume audio-visual streams. As such, either the content provider must support an increasing number of versions of the same content or the consumer must be satisfied with content that is less suited for his particular context.

The *Universal Multimedia Access* (UMA) framework provides an answer for these problems. The core idea is to enable the consumption of multimedia content for different usage contexts by creating different representations of the same information from a single content base. In other words, the UMA framework adheres to the "create once, play everywhere" paradigm.

The objective of this thesis is to investigate into the requirements for

realizing a UMA-compliant architecture and application, for example a Video-on-Demand application, and to address various issues during the creation.

A first part that is investigated is how to describe in a standardized way *what* is being consumed – i.e., the *content* – and *how* this is being consumed – i.e., the *context*. In order to be able to compare the various international standards for content description, an objective evaluation and comparison framework is created which allows one to determine the most optimal standard for a particular usage. As a result, the *MPEG-7* specification proves to be a good selection to use in the UMA-compliant architecture. Next, three context-description standards are studied. From these, the *MPEG-21 Digital Item Adaptation – Usage Environment Description* standard proves to be the most generic and comprehensive standard available for context description. Although this specification is vast, a software toolkit is developed that runs on very constrained devices (such as cell phones) and that is able to read, modify, and write MPEG-21 compliant messages.

The content and context information is used by a *content adaptation engine* to optimize the audio-visual content in such a way that the resulting optimized version is consumable in the given context. This thesis does not focus on determining how the content adaptation should be performed, but researches the prerequisites in order to make such a decision. In other words, it is investigated how the content and context information can be *negotiated* with the decision-taking engine by invoking a web service. An important extension to the UMA framework is introduced, namely the *time-varying metadata* concept. The main idea is to dynamically optimize the content to a changing context by re-negotiating the context information. Modifications to the consumption environment result in the re-optimization of the content on-the-fly.

The adaptation of the audio-visual streams is another important part in this thesis. After an overview of the current state of the art in video scalability techniques and standards, special attention is paid to the *Region-of-Interest* (ROI) concept. A ROI is an area within a video that is seen as more important than the remaining area. MPEG-4 FGS makes it possible to visually improve this area. Unfortunately, this is a fixed region throughout the videos sequence. New and *lightweight object tracking techniques* are introduced that work within the compressed domain by reusing the motion vector field to overcome this shortcoming. Several tests are performed to demonstrate the quality of my object

tracking scheme and a time-complexity analysis proves that the scheme is fast enough to work in real-time. Although the object tracking is implemented in MPEG-4 FGS, the algorithms are generic and can be used within other video codecs.

Furthermore, an important issue is solved, namely the *verboseness* and lack of support for *updates* of XML-based data, such as the content and context information. First, the techniques to process XML-based data are investigated and modeled. Based on this research, a *serialization-agnostic XML* parser is created. Applications using this parser can handle XML data without being aware of the actual encoding format. So, it is possible to use a non-verbose (binary) marshalling of the XML data instead of traditional plain-text serialization. Three techniques (ZIP compression, ASN.1-PER, and MPEG-B BiM) are studied as potential alternatives by evaluating their compression efficiency (hence, overhead reduction) and their usability in applications. The evaluation proves that the BiM technology is the best for overhead reduction, however not yet usable due to its complexity. Although ZIP compression does not achieve the highest compression ratio, it can be used immediately as alternative XML marshalling format.

Finally, all previously discussed techniques are integrated in a Video-on-Demand application that is compliant with the UMA framework and that can handle time-varying metadata. As such, I hope to have convinced the reader that the techniques introduced in this thesis are useful to create and to ameliorate (Internet-based) multimedia applications and help to simplify the construction of genuine UMA-compliant applications.

List of Abbreviations

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
AVC	Advanced Video Coding
BBC	British Broadcasting Corporation
BER	Basic Encoding Rules
BiM	Binary MPEG Format for XML
CC/PP	Composite Capability / Preference Profiles
CER	Canonical Encoding Rules
CIF	Common Intermediate Format
CLDC	Connected Limited Device Configuration
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DCMES	Dublin Core Metadata Element Set
DCMI	Dublin Core Metadata Initiative
DCOM	Distributed Component Object Model
DCT	Discrete Cosine Transform
DDL	Description Definition Language
DER	Distinguished Encoding Rules
DIA	Digital Item Adaptation
DOM	Document Object Model
EBU	European Broadcasting Union
ERD	Entity Relationship Diagram
ESCORT	EBU System of Classification of Radio and Television Programs
FGS	Fine-Granularity Scalability
GOP	Group Of Pictures
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HDTV	High Definition Television
HTTP	Hypertext Transfer Protocol
IEC	International Engineering Consortium
IP	Internet Protocol
IPMP	Intellectual Property Management and Protection

ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU.T	ITU Telecommunication Standardization Sector
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JVM	Java Virtual Machine
JVT	Joint Video Team
KLV	Key - Length - Value
LDAP	Lightweight Directory Access Protocol
LSB	Least Significant Bit-plane
LZW	Lempel - Ziv - Welsh
MARC	Machine-Readable Cataloging
MC-3DSBC	Motion-Compensated Three-Dimensional Subband Coding
MC-EZBC	Motion-Compensated Embedded Zerotree Block Coding
MCTF	Motion-Compensated Temporal Filtering
MDS	Multimedia Description Schemes
METS	Metadata Encoding & Transmission Standard
MIDP	Mobile Information Device Profile
MIME	Multipurpose Internet Mail Extensions
MOA2	Making of America II
MSB	Most Significant Bit-plane
MSDN	Microsoft Developer Network
MPEG	Moving Picture Experts Group
OSI	Open System Interconnection
PAL	Phase Alternating Line
PER	Packed Encoding Rules
PPM	Prediction by Partial Match
QCIF	Quarter CIF
RDF	Resource Description Framework
ROI	Region-of-Interest
RPC	Remote Procedure Call
RSS	Really Simple Syndication
RTP	Real-Time Transport Protocol
RTCP	Real-Time Control Protocol
RTSP	Real-Time Streaming Protocol
SAX	Simple API for XML
SDTV	Standard Definition Television
SMEF	Standard Media Exchange Framework
SMTP	Simple Mail Transfer Protocol
SNR	Signal-to-Noise Ratio
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
STX	Streaming Transformations for XML
SVC	Scalable Video Coding
TCP	Transmission Control Protocol

UAProf	User Agent Profile
UDDI	Universal Description, Discovery, and Integration
UED	Usage Environment Description
UMA	Universal Multimedia Access
UMID	Unique Material Identifier
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UTF-8	8-bit Unicode Transformation Format
VCEG	Video Coding Experts Group
VLC	Variable Length Coding
VoD	Video-on-Demand
VOP	Video Object Pane
VRT	Flemish Radio- and Television Network
W3C	World Wide Web Consortium
WAP	Wireless Access Protocol
WBXML	WAP Binary XML
WSDL	Web Services Description Language
XER	XML Encoding Rules
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Contents

Chapter 1

Introduction

1.1 Context

During the last decade, the Internet steadily became a familiar technology as more and more users enjoy its benefits day-to-day. Many new and different kinds of (mobile) end-user devices enable ubiquitous access. As users are more acquainted with the possibilities of the Internet, they increasingly demand full access. They are satisfied with a rich multimedia experience at home, but they also desire this experience anywhere, anytime, on any device.

To date, ad hoc solutions are in place to provide the end user with a rich multimedia experience, typically intended for a particular usage on a particular device for a particular network. A well-known solution is to store semantically equal multimedia data several times with different characteristics, for example different resolutions, in order to address the different end-user devices. However, this method is no longer sustainable as the number of different types of end users, end-user devices, networks, and content rapidly increases. Assume, an end user wants to consume an audio-visual stream. The success of this elementary operation depends on many factors:

- The *end user* is a first, but often forgotten, important factor. Indeed, his preferences determine, for example, the language of the audio stream; his age implies whether or not he is authorized to see particular content; and information on his hearing or visual deficiencies can help to optimize the content. In other words, the

end user decides how the content is presented. Content providers should try to indulge these wishes because "the customer is always right."

- Different *end-user devices* have very diverse characteristics and possibilities. Examples are the screen size, the resolution, the (remaining) battery capacity, the processing speed, the supported video decoders, and so on. On a regular basis, a new class of devices becomes available with features distinct from the existing ones. It is not ideal, and usually impossible, to use the same content for two different classes of devices. For example, a cell phone is not able to process content optimized for a high-end multimedia desktop computer.

- The *network* is a third important factor. Thanks to network protocols that are constructed according to the *Open System Interconnection* (OSI) network layer model[1], applications – such as an audio-visual content streaming service – are shielded from the technical low-level details of a network connection. Nevertheless, the specific kind of network technology, protocol, and carrier influences network delay, packet loss, error rate, and so on. Still the most important limiting network factor to take into consideration for audio-visual content streaming is the available bandwidth. Although new and high-speed technologies become available and are affordable, relatively slow networks are still being used and even complemented with new technologies for specific market segments, such as *General Packet Radio Service* (GPRS) for cell phones. As such, the range in bandwidths that content providers must support is still increasing.

- The *content provider* is the next factor to take into account as he is responsible to *stream* the requested data to the end-user devices. As such, he decides how the audio-visual data are encoded and which network protocols are used. On top of that, if the number of clients increases, the content provider must divide its saturated bandwidth over several streams which results in a reduction of the bandwidth for the individual streams. Furthermore, the processing capacity of the content provider must be adequate to handle all the requests of the clients.

[1]The OSI model is a standard of the *International Organization for Standardization* (ISO) and defines a networking framework in seven layers. More information on OSI is available at `http://www.iso.org`.

- Previous factors deal with the context of the usage, the final factor deals with the *content* itself. The information about the context can be used to determine the content layout. For example, if the audio-visual stream is intended to be consumed on a PDA with a display resolution of 352×288 pixels, the content should fit this resolution. This can be accomplished by creating multiple representations with different resolutions of the same content, however this *simulstore* results in storage overhead. Another solution is storing content in a *scalable* way. This means that different versions can be derived from the scalable bitstream on-the-fly by simple, usually truncating, operations.

To cope with all these factors, a huge amount of ad hoc solutions must be created. On top of that, some factors change during the consumption of the multimedia content, such as the available bandwidth. This can imply that the initial ad hoc solution is no longer appropriate and that a new solution must be used or, in a worst case scenario, generated during the actual consumption of the audio-visual content.

1.2 Goal and Outline

The goal of my thesis is to investigate into the requirements making multimedia consumption possible anywhere, anytime, and anyhow by adhering to the *Universal Multimedia Access* (UMA) concept. This concept states that UMA-compliant architectures enable the consumption of multimedia content for different usage contexts (hence, different end-user interests and profiles, end-user devices, networks, and content providers) by creating different representations of the same information from a single content base [1].

In other words, in a UMA-compliant architecture the multimedia content is optimized to what the end user wishes, while considering the context wherein the content is consumed. As such, UMA subscribes the "create once, play everywhere" paradigm.

In this thesis, I look at the different aspects that are needed in order to create such an architecture according to the UMA principles.

In Chapter 2, a study is made on how to describe the content and the context in a standardized way. For the former, I have developed evaluation criteria to perform an objective comparison of the many available

audio-visual content-description standards. For the latter, I have created a software tool, based on a *Moving Picture Experts Group* (MPEG) standard, to express and handle information about the context. This tool works on most end-user devices, even in very constrained environments like on a cell phone.

The information about the content and the context is used to optimize the multimedia content. This optimization is done by an *adaptation engine*. In Chapter 3, different strategies on the location of this engine in a UMA architecture are discussed. Wherever it is located, the content and context information must be *negotiated* with the engine. I will discuss the different possibilities on how to do this. I further elaborate on the necessity that context updates – such as a change in the available bandwidth – must be supported, which is my novel contribution to the UMA concept. To conclude this chapter, I will demonstrate that, because the *Extensible Markup Language* (XML) is used to record the content and context information, there is overhead and hence bandwidth is wasted during the negotiation phase.

In Chapter 4, I solve this overhead issue using alternative serialization methods instead of the classical plain-text notation of XML-based data. I create an XML parser that is capable of handling any kind of serialization technology in such a way that the user (for example, an application) is unaware of the used serialization format. My solution is evaluated by applying it to the context information that is negotiated with the adaptation engine. To further illustrate the advantage of my approach, I applied the system to a well-known XML-based Internet application, namely the *Really Simple Syndication* (RSS).

Chapter 5 is about video scalability as a means for multimedia content optimization. After an overview of the different types and ways to create scalable video content, I introduce a novel and fast technique with low time complexity for object tracking. The feasibility of my technique is demonstrated by implementing the algorithms in a scalable video encoder. The results of my object tracking algorithms are compared to the results of a manual tracking scheme performed by a test audience.

Chapter 6 discusses the integration of the various techniques from the previous chapters in a UMA-compliant *Video-on-Demand* (VoD) application with support for context information updates. Finally, concluding remarks are drawn with an overview of the major contributions and results of my research in this thesis.

1.3 Overview Publications

The research that has lead to this thesis resulted in a number of publications. Two papers are accepted for publication in journals that appear in the Science Citation Index, namely in Springer's *Multimedia Systems* [2] and in Springer's *Lecture Notes in Computer Science* [3]. Two papers are currently under review for publication in journals that appear in the Science Citation Index, namely Eurasip's *Journal on Applied Signal Processing* [4], and Elsevier's *Journal of Visual Communication and Image Representation* [5]. In addition, I have contributed 11 papers to international conferences as first author [6–16]. Collaboration with fellow researchers resulted in 14 publications as co-author [17–30]. Finally, 10 contributions were submitted to the MPEG community [31–40].

Chapter 2

Metadata

2.1 Introduction

As outlined in the introduction of this thesis, we want to investigate the requirements to construct a UMA-based architecture. A first element required for this construction is the need to have information on "what" is being played – i.e., the *content* – and "how" it is being played – i.e., the *context*. Assume the following straightforward example: a user wants to consume high-resolution content (e.g., a video stream with a resolution of 1920×1080 pixels) on his PDA that has a low-resolution display (e.g., 352×288 pixels). The content and context information informs us that the content must be adapted, in particular scaled-down, to make consumption possible. Additional information on the content can give hints to an *adaptation engine* how this adaptation can be executed. Likewise, additional information on the context can result in additional constraints.

In this chapter, we study the prerequisites for content adaptation, namely the required information about the content and the context to perform the adaptation. A generalized term for this supplemental information is *metadata*. In this chapter, we discuss the different technologies to describe content and context. The latter encompasses metadata about the network, the end user and his preferences, the end-user device, the content provider, and so on. The former includes metadata about the type of content, the compression (encoding and decoding) technique, rights information, et cetera.

First, we investigate how to create, structure, and denote the metadata for content information in Section 2.2. We start by investigating the existing technologies used by the larger content creators and consumers, namely the television broadcasters. However, their techniques prove to be insufficient for the description of audio-visual content, which is the kind of content we envisage in the UMA-compliant architecture. Luckily, many new international standards are available, specifically for content annotation. To compare them, we have defined different evaluation criteria. These criteria allow the selection of the most optimal specification for a particular application. As an example, four international standards are discussed and compared by our criteria, namely, *Dublin Core Metadata Element Set* (DCMES), *Multimedia Content Description Interface* (better known as *MPEG-7*), *P/Meta*, and *Standard Media Exchange Framework* (SMEF).

Next, we examine the techniques that can be used to describe the context information in Section 2.3. Three technologies are discussed in detail, namely *HTTP Headers*, *Composite Capability / Preference Profiles* (CC/PP), and the *MPEG-21 Part 7 – Digital Item Adaptation – Usage Environment Description* (MPEG-21 DIA-UED) tool. Finally, we evaluate the usability of the latter, even for constrained devices such as cell phones. Our software developed for this purpose is currently accepted by the MPEG consortium as the reference software tool for the MPEG-21 DIA-UED technology [11, 32].

2.2 Metadata for Content Description

We have investigated the methods that are currently used by the larger creators and consumers of (audio-visual) content, namely the television broadcasters. As they work day-to-day with new and archived audio-visual material, systems must be in place to find and retrieve the desired data. In other words, television broadcasters are most likely to have good content-description systems available.

Although many international standards are available, an audit of several Flemish television broadcasters' audio-visual libraries demonstrates that each broadcaster uses its own proprietary system to annotate their (currently mostly tape-based) collections. As a result, exchanging content between broadcasters and opening up these collections to consumers is complicated. Nevertheless, most broadcasters desire such an interopera-

ble library of audio-visual content, which is preferably a digital file-based system.

In [41] it is observed that "the metadata necessary for successful management and use of digital objects is both more extensive than and different from the metadata used for managing collections of physical material." In other words, the metadata used to annotate tape-based collections are not sufficient for digital file-based libraries. Hence, the television broadcasters themselves are currently investigating which international standard to implement. However, they are overwhelmed by the extensive choice.

2.2.1 Evaluation Criteria

To address the aforementioned issue, we have defined several selection criteria that can be used to compare and evaluate different metadata standards for content description. These criteria are composed in such a way that all aspects ranging from content organization to the different types of metadata are taken into account. Nevertheless, the criteria are independent of any restriction imposed by a particular content management system.

Criterion 1: internal vs. exchange standards

For this first criterion, it is important to identify the involved parties that exchange audio-visual content during its typical life cycle. The *European Broadcasting Union* (EBU) identifies in [42] the consumers and three trading entities, being the content creator, the content distributor, and the archive. EBU has investigated the different relationships between these four players and has presented the entities and the relationships in the EBU P/Meta Business-to-Business Dataflow Model (see Figure 2.1). This model is independent of any content-description standard and is applicable for most content providers.

On the one hand, particular standards are specifically developed for managing the metadata in the interior of a system. These standards are further referred to as *internal standards*. Usually, they are represented as an *Entity Relationship Diagram* (ERD) and describe the architecture of the database that stores the metadata of the audio-visual content.

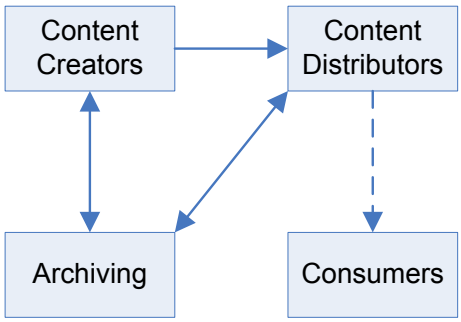


Figure 2.1: EBU P/Meta Business-to-Business Dataflow Model [42].

On the other hand, other standards are used to structure the information about the content to make transmissions between the different parties possible. We call these standards *exchange standards*. "Exchange" must be seen as broad as possible, namely between any combination of content creator, content distributor, archive, and consumers.

Criterion 2: flat vs. hierarchical standards

The structural organization of the content description is a second criterion. This indirectly determines how fine-grained the content is described. Two extreme visions can be identified. On the one hand, the content is considered an elementary and indivisible unit, resulting in a coarse description, and on the other hand, the content is divided in small sub-pieces each annotated separately, resulting in a very fine-grained description.

If the content is considered as an elementary and indivisible unit, the content provider can associate this elementary unit with, for example, a *program*. The metadata describes the content as a whole and does not describe its individual parts. This model is referred to as a *flat standard*.

Sub-parts of the content can be annotated with much more detail. The additional metadata belongs to the individual parts and permits the user to perform a more detailed search on the content. For example as illustrated in Figure 2.2, a program can be split up in several *editorial objects*, corresponding to the individual scenes. Every editorial object can be annotated with additional descriptive metadata, so it is possible to search on the editorial object itself. In turn, editorial objects can

be broken down in different *media objects*. These media objects could be the audio components, the video components, the subtitles, and so on. This is also possible the other way around: a group of programs belonging together can be collected in a *program group*. This program group is annotated with information identical to all programs it holds, for example, the name of the program. Hence, it is not necessary to repeat the same information for every program, but the program inherits information from its program group. The underlying idea is that information has to be added to the objects at the right level. This concept is referred to as a *hierarchical standard*. A four-layered architecture as discussed above is visualized in Figure 2.2.

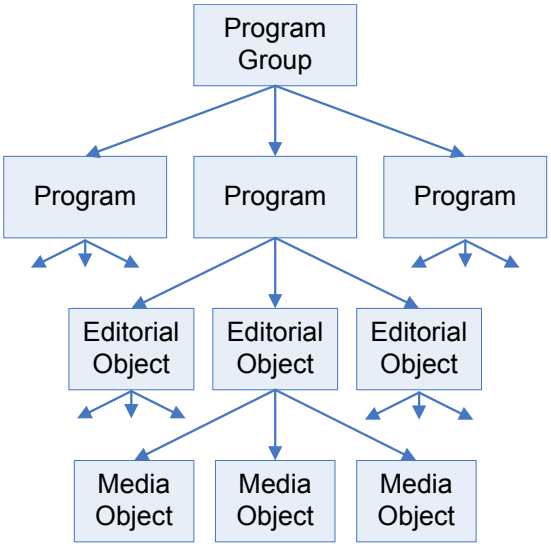


Figure 2.2: Hierarchical organization of content description.

The content provider will not always want to use a hierarchical standard, although this has huge benefits for faster and more efficient search and retrieve operations. Indeed, the most important reason for a content provider to restrict the metadata (and thus the decomposition of the audio-visual asset), is to limit the cost associated to the amount of metadata that needs to be collected. It is clear that, as the metadata about an audio-visual object grows, the marginal profit of the additional metadata decreases, but the cost to generate this additional metadata increases disproportionally. At a certain moment, it will be impossible to add additional metadata without making unjustified costs. In other

words, the content provider will have to make a trade-off between cost and comprehensiveness of the description of the content.

Criterion 3: supported types of metadata

Digital content is useless without technical information because we cannot decode it. Adding rights information can augment content into an asset. These are two examples of metadata types. Similar to the trade-off between cost and comprehensiveness with regard to the granularity of content description of criterion 2, content providers will have to select the types of metadata they want to support. We can identify five types as follows:

- *Identification Metadata:* the identification metadata are primarily about information to singularly identify content. This can be done by human interpretable fields, like a title or an index, or by machine understandable identifiers, like a *Unique Material Identifier* (UMID) or a *Uniform Resource Identifier* (URI). Besides the identification metadata related to the content itself, other identifying information could be required to locate content-related documents that are potentially stored in another system, for example, a contract number.

- *Description & Classification Metadata:* the descriptive metadata describes what the content expresses. This could be done by providing a list of keywords that try to place the content in a particular semantic context. In some cases, the keywords are selected from an organized dictionary of terms, i.e., a *thesaurus*. Other classification schemes can be used to categorize the content in different pre-defined classes, such as the genre and the audience. A very well-known classification system is the *EBU System of Classification of Radio and Television programs* (ESCORT) 2.4 system [43] that organizes the content in conceptual, administrative, production, scheduling, transmission, viewing, and financial groups. Another type of descriptive metadata comprises the description of the content as short free-form text. This type of descriptive metadata is well known and is extensively used in practically every content annotation system. Unfortunately, this kind of field is error-prone (e.g., spelling mistakes) and should be used with care.

- *Technical Metadata:* the technical metadata describes the technological characteristics of the related content. The minimal required technical metadata must specify the audio and video decoder. This minimal information gives the users the possibility to consume the content. Hence, the technical metadata enables the content to become usable.

- *Security & Rights Metadata:* the security metadata handles all aspects from secure transmission (i.e., the encryption method) to access rights (i.e., who has access clearance). As such, content might become an *asset* for the content provider. The access rights metadata can be split up in information about the rights holder and information about contracts. The rights holder is the organization who owns the rights of the audio-visual content. Also, the contracts related to the publication of the content and the contracts of the people who are involved with the creation of the content, are considered as rights metadata.

- *Publication Metadata:* the last type of metadata describes the publication(s) of the content. Every publication establishes a date of publication, the time and duration of the publication, the method of publication, and so on. This way, the content provider has an idea of the frequency and the popularity of the content. Furthermore, this information is important to clear publication rights and handle payments.

Criterion 4: syntax & semantics

Some standards define only syntax, others only semantics, but most define both. The syntax defines how the representation of the metadata must be done so a choice must be made between a textual and a binary representation. The textual representation has the advantage that the metadata are human readable, but at the same time it is very verbose. The binary representation is dense, but it has the disadvantage that it can only be handled by machines.

In case of plain-text notation, XML [44] is mostly used. If so, the metadata standard provides, besides the standard itself, usually an *XML Schema* [45] that rigorously determines the syntax of the metadata. This makes it possible to check the correctness (i.e., *validity*) of the metadata. XML data can also easily be converted to another form, e.g.,

XML data that is valid to another XML Schema, using a transformation style based on a transformation language such as the *Extensible Stylesheet Language Transformation* (XSLT) [46] or *Streaming Transformations for XML* (STX)[1]. These XML characteristics enable interoperability. More information on XML and its various tools is available at the *World Wide Web Consortium* (W3C) Website[2].

The semantics of the metadata standard determine the meaning of the metadata elements. Without any semantic description, one is free to assume the meaning of the different metadata elements, presumably resulting in different interpretations between users. Only if the description of the metadata elements is strictly defined, all users must agree on the meaning of the metadata elements, which improves interoperability. The former is hereafter referred to as *open* semantics; the latter is referred to as *closed* semantics.

2.2.2 Content-Description Standards

In this section, we apply our evaluation criteria to four well-known metadata standards, namely DCMES, MPEG-7, P/Meta, and SMEF. Table 2.1 gives a schematic overview of this evaluation of the four metadata standards. The results are used in the next section to determine the optimal content-description standard to use in a UMA architecture.

Table 2.1: Overview of the Evaluation of the Metadata Standards.

	DCMES	MPEG-7	P/Meta	SMEF
criterion 1	exchange	exchange	exchange	internal
criterion 2	flat	hierarchy	hierarchy	hierarchy
criterion 3	identification, description, and technical (limited)	identification, description, and technical	all	all[†]
criterion 4	open XML & RDF *	closed XML	closed XML	closed ERD

[†] SMEF emphasizes rights metadata.

* DCMES can be mapped to XML and RDF.

[1] More information on STX is available at `http://stx.sourceforge.net`.
[2] The W3C Website is available at `http://www.w3.org`.

Dublin Core Metadata Element Set

The *Dublin Core Metadata Initiative* (DCMI)[3] is an open consortium engaged in the development of interoperable and online metadata standards that support a broad range of purposes and business models. The DCMI defined in 1999 the first version of the Dublin Core Metadata Element Set that consists of 15 elements. In a second phase, this model was extended with three additional elements and a series of refinements, resulting in version 1.1 of the specification [47].

The goal of the DCMES specification is to exchange resource descriptions aiming at cross-domain applications (*criterion 1*). Both versions of the specification are very straightforward and have two very important limitations. On the one hand, there are no provisions for describing hierarchically structured audio-visual content – however, this can be circumvented by making implicit references to other parts, – hence DCMES is a flat standard (*criterion 2*). On the other hand, the number of available metadata elements is too limited for thoroughly annotating audio-visual resources (*criterion 3*).

With regard to *criterion 4*, the semantics are concisely described, still the user has considerable freedom for his own interpretation. The DCMI provides different ways for syntactically describing the metadata: there are guidelines for incorporating DCMES in XML[4] and guidelines to use it in combination with the *Resource Description Framework* (RDF)[5] [48].

Multimedia Content Description Interface

The *International Organization for Standardization* and the *International Engineering Consortium* (ISO/IEC) have created the International Standard 15938, formally named *Multimedia Content Description Interface*, but better known as the MPEG-7 standard, which provides a rich set of tools for thoroughly describing multimedia content [49–51].

The MPEG-7 standard is developed for, among other things, the exchange of metadata describing audio-visual content. It has been de-

[3] More information on DCMI is available at `http://dublincore.org`.

[4] The guidelines for the notation of DCMES in XML format are available at `http://dublincore.org/documents/dc-xml-guidelines`.

[5] The guidelines for the notation of DCMES in RDF format are available at `http://dublincore.org/documents/dcq-rdf-xml`.

signed to support a broad range of applications, without targeting a specific application. As such, it is an exchange standard (*criterion 1*).

The MPEG-7 standard normatively defines the syntax, using an XML Schema, and the semantics, via normative text, of all metadata elements (*criterion 4*). The elements are structured as *descriptors* and *description schemes*: a descriptor represents a particular feature of the audio-visual content; a description scheme is an ordered structure of both descriptors and other description schemes. This system is used to create a hierarchy (*criterion 2*). For example, the audio-visual material can be described by its temporal decomposition and by its media source decomposition. The latter is divided into descriptions about the audio segment and the video segment. This video segment is further decomposed into shots, key frames, and objects.

The supported types of metadata (*criterion 3*) are mostly focused on the description, technical, and, to a lesser degree, identification metadata. No attention was paid to publication and rights and security metadata elements, ISO/IEC addresses these types of metadata in different parts of the MPEG-21 standard [52, 53].

MPEG-7 consists of several parts: part three and four deal with the technical metadata for video and audio content respectively; part five, also referred to as *Multimedia Description Schemes* (MDS), defines descriptors and description schemes for the classification of audio-visual content. More information about MDS is given in [54] and an overview of its different functional areas is visualized in Figure 2.3.

P/Meta

The P/Meta standard is developed by the EBU as a metadata vocabulary for program exchange in the professional broadcasting industry [55, 56]. Hence, it is not intended as an internal representation but as an exchange format for program-related information in a business-to-business environment (*criterion 1*).

The P/Meta standard creates a five-layered hierarchy (*criterion 2*): the brand, the program group, the program, the program item, and the media object. A brand collects all the program groups with a recognizable collective identity, e.g., information about the broadcasting station. Every program group is composed of individual programs, which consist of individual program items. Finally, every program item may be split up

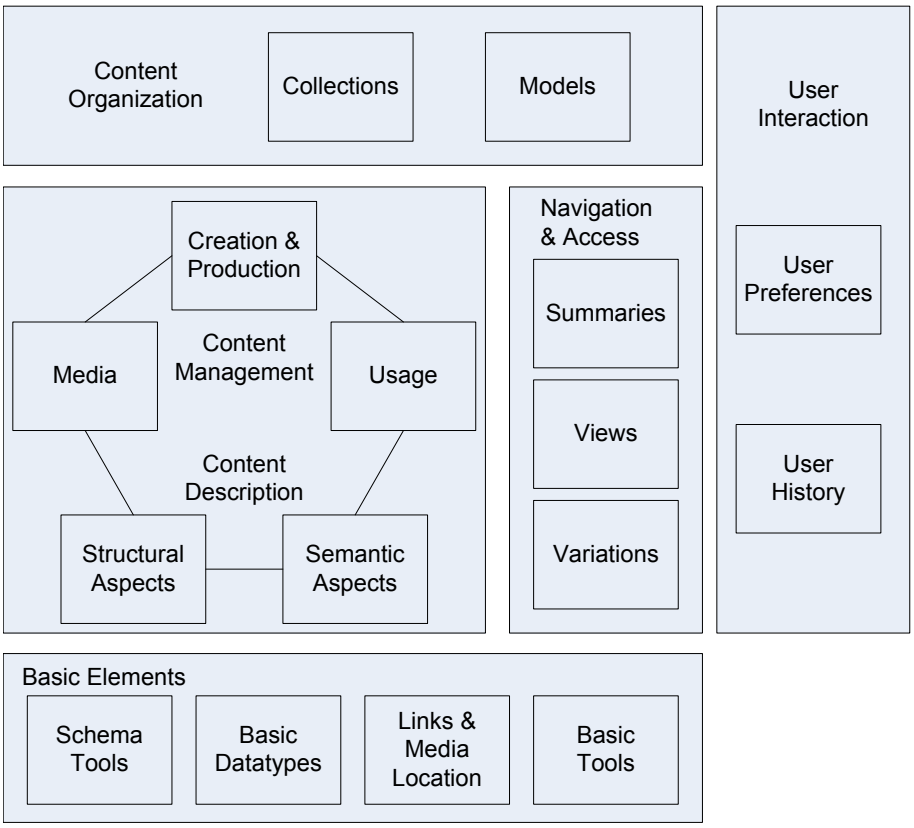


Figure 2.3: Overview of MPEG-7 Part 5 – Multimedia Description Schemes [54].

in media objects. This hierarchy is comparable with the one illustrated in Figure 2.2.

To obtain this hierarchical structure, the standard defines a number of *P/Meta Sets* and *P/Meta Attributes*. The latter is an element describing a particular feature; the former groups P/Meta Attributes and other P/Meta Sets in such a way that all relevant metadata are collected for describing the considered object. For example, every program group and every program is annotated with identification (numbers and titles), classification (according to the ESCORT 2.4 system), and descriptive metadata. Besides these three elementary types, the description of the individual programs is complemented with four types, namely transmission or publication metadata, metadata concerning editorial objects

and media objects, technical metadata (audio and video specification, compression schemes, and so on), and rights metadata (contract clauses, rights list, and copyright holders). These enumerated types are also the supported types of metadata (*criterion 3*).

P/Meta strictly defines all sets and attributes, resulting in a metadata standard where every term is determined unambiguously. The syntax is defined by an XML Schema (*criterion 4*).

Standard Media Exchange Framework

The Standard Media Exchange Framework has been developed by the Media Data Group of BBC Technology, now Siemens SBS, on behalf of the *British Broadcasting Corporation* (BBC). Through a close collaboration with a wide variety of BBC projects, a profound understanding of the broadcaster's audio-visual media information requirements has been derived. Although the model is developed for use within the BBC, the definitions are organization independent and are usable for any other (large) broadcaster. Also, SMEF is not endorsed by any standardization organization. Nevertheless, this framework is seen by other broadcasters as a reference model.

SMEF[6] provides a rich set of data definitions for the range of information involved in the production, development, use, and management of media assets. Its purpose is to ensure that different in-house systems store this information in the same way. Therefore, the SMEF standard defines an ERD that provides a framework for storing the metadata in the system. It is intended to be used within one broadcaster. Although the metadata may flow between different internal division, we regard this specification as an internal standard (*criterion 1*).

The SMEF metadata model records all information that becomes available during the whole production cycle, from a program concept over media and editorial objects to the actual publication. An editorial object (e.g., a program) can be split up in different media objects, making this a hierarchical metadata model (*criterion 2*). Each media object can be annotated extensively with descriptive and technical metadata. The editorial object and media object entities can be linked with two other entities, namely the usage restriction entity (describing the restrictions

[6]More information on SMEF is available at `http://www.bbc.co.uk/guidelines/smef`.

on the use) and the copyright reporting entity (describing copyright details of the material). Hence, SMEF pays much attention to the rights metadata (*criterion 3*).

With regard to *criterion 4*, the SMEF standard defines the semantics of all entities and relationships. The definition of syntactical rules covers the way the metadata are represented in the internal system.

2.2.3 Selecting Our Content-Description Standard

The selection of the optimal content-description standard depends on the intended usage. In this section, we determine the optimal standard for a UMA-compliant application that streams audio-visual content over a network to a consumer. This content is optimized by a content adaptation engine to suit the consumption context. Table 2.2 gives an overview of the requirements for the content-description standard to be used in the UMA architecture.

Table 2.2: Requirements Content-Description Standard.

Criteria	Requirement
criterion 1	exchange
criterion 2	n/a (flat or hierarchy)
criterion 3	technical metadata
criterion 4	closed & XML

Our UMA-based use case implies that we require an exchange standard as the content information will be handled by an adaptation engine. This adaptation engine can be located anywhere between the content provider and the content consumer, as will be explained in the next chapter. As a result, the content description must be transmitted to this engine, hence the selection of an exchange standard.

The description of the content is processed by a content adaptation engine. How this information is structured is therefor not a concern and can be flat or hierarchical.

With regard to the types of metadata, we especially need extensive technical information in order to assess the possibilities of content adaptation. Descriptive metadata can also be used to steer content adaptation (for example, to enable content summarization or to delete violent

scenes). Identification metadata are less important for content adaptation and, strictly speaking, we do not need the security and rights information nor publication metadata to perform content adaptation.

To improve interoperability, the content-description standard should have a closed semantic and a strictly defined syntax, for example by an XML Schema.

Given these requirements and the evaluation of the standards in the previous section, we can conclude that MPEG-7 and P/Meta are both candidates. On the one hand, P/Meta contains more types of metadata than MPEG-7. On the other hand, MPEG-7 contains more low-level (technical) descriptions of the content than the broadcaster's oriented P/Meta. Because the low-level content descriptions facilitate content adaptation, MPEG-7 has a slight preference over P/Meta and the other standards as the preferred standard to describe audio-visual content for UMA-based applications. Note, the content-description standard might be different for other (non-UMA) applications.

2.3 Metadata for Context Description

In this section, we investigate the available techniques to annotate the context of the content consumption. Like the description of content, it is important to use a non-proprietary, well-defined, and preferably standardized solution in order to ensure interoperability. Unfortunately, only a few standardized vocabularies to describe the context are available today. In this section, we discuss the three most important ones, namely the use of *HTTP Headers*, *Composite Capability / Preference Profiles*, and the *MPEG-21 Digital Item Adaptation – Usage Environment Description* tool.

The latter is the most generic and extensive one. To validate its usability for constrained devices, we have developed a toolkit that is capable to read, process, and write MPEG-21 DIA-UED compliant messages. This toolkit is explained at the end of this section.

2.3.1 Context-Description Standards

HTTP Headers

The importance of having information about the usage context can already be seen in the first version of the *Hypertext Transfer Protocol* (HTTP) as HTTP/1.0 [57] has a limited provision to exchange context information between an end-user device (usually a Web browser) and a content provider (always a Web server). This is accomplished by adding a `User-Agent` field to the HTTP Header that contains information about the requesting application; usually this is the name and version of the Web browser such as "Mozilla/4.0."

HTTP/1.1 [58] extends the context description capabilities of the HTTP Headers by defining the following additional header fields:

- `Codec Capabilities`: specifies all encoding and decoding capabilities of the end-user device.
- `Accept`: this header field is used to specify the media types that can be handled by the client application. The client application lists the *Multipurpose Internet Mail Extensions* (MIME) types it can process. Examples are `text/html` and `audio/wav`.
- `Accept-Charset`: the client uses this field to inform the server about the character encoding set it can handle, for example, `us-ascii`, and `iso-8859-5`.
- `Accept-Encoding`: similar to the `Accept`-header, but it is stricter as the client only accepts the enlisted media types.
- `Accept-Language`*:* lists the preferred language(s) (and optionally the country code) of the client application, for example, `nl-BE`, `en-US`, and `en`.
- `From`: contains the e-mail address of the end user.
- `Pragma`: this header field contains application-specific additional information.

Listing 2.1 shows an example on the usage of these fields.

Listing 2.1: HTTP Headers request example.

```
GET / HTTP/1.1
Host: www.ugent.be
Connection: close
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
     application/x-shockwave-flash, */*
Accept-Charset: us-ascii
Accept-Encoding: gzip
Accept-Language: en,nl-be
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
    5.1; SV1; .NET CLR 1.1.4322; InfoPath.1)
```

Due to its nature, this solution is tightly coupled to the consultation of Web pages. As such it is less suitable to use in other situations, like requesting a video stream. Also, the description of the usage context is very limited, however, it can be extended by adding application-specific `Pragma` headers, but this jeopardizes interoperability.

Composite Capability / Preference Profiles

CC/PP [59] is a W3C recommendation of its *Device Independence Activity* group. The task of this group is "to allow access to the web by any Internet-enabled device." CC/PP is a framework that enables the creation of *CC/PP profiles* which describe device capabilities and user preferences. It does not define a vocabulary itself, but allows a third party to use this framework to create one.

A CC/PP profile is structured as a two-level hierarchy:

- *Main components*: a major branch in the profile, such as a *hardware* and a *software* component.
- *Attributes*: detailed information logically associated with a component. For example, *screensize* as part of the hardware component.

The combination of the components and their attributes creates the vocabulary of a CC/PP profile. Apart from the vocabulary, the profile defines its syntax using RDF, either in graph notation or by an XML Schema.

Notwithstanding the flexibility of the framework, only a few CC/PP profiles are created to date, such as the *User Agent Profile* (UAProf) [60]

defined by the Open Mobile Alliance [7], formerly known as the *Wireless Application Protocol* (WAP) Forum. This profile defines six main components – hardware, software, network, browser, WAP characteristics, and push characteristics, – each with various attributes. It is mainly intended to describe the capabilities of cell phones in a standardized way. An excerpt of the Nokia 3650 cell phone's profile can be found in Listing 2.2[8].

Listing 2.2: UAProf of the Nokia 3650 cell phone (excerpt).

```
<?xml version="1.0"?> <rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:prf="http://www.openmobilealliance.org/[...]/ccppschema-20021212#">
 <rdf:Description rdf:ID="Nokia3650">
   <prf:component>
    <rdf:Description rdf:ID="HardwarePlatform">
     <prf:ScreenSize>176x208</prf:ScreenSize>
     <prf:Model>Nokia3650</prf:Model>
     <prf:ScreenSizeChar>15x6</prf:ScreenSizeChar>
     <prf:ColorCapable>Yes</prf:ColorCapable>
     [...]
    </rdf:Description>
   </prf:component>
   <prf:component>
    <rdf:Description rdf:ID="SoftwarePlatform">
     <prf:SoftwareNumber>6.1</prf:SoftwareNumber>
     <prf:JavaEnabled>Yes</prf:JavaEnabled>
     <prf:JVMVersion><rdf:Bag><rdf:li>
         SunJ2ME1.0
     </rdf:li></rdf:Bag></prf:JVMVersion>
     <prf:OSName>Symbian OS</prf:OSName>
     [...]
    </rdf:Description>
   </prf:component>
   <prf:component>
    <rdf:Description rdf:ID="NetworkCharacteristics">
     <prf:SupportedBearers>
      <rdf:Bag>
      <rdf:li>GPRS</rdf:li>
      <rdf:li>CSD</rdf:li>
      </rdf:Bag>
     </prf:SupportedBearers>
     [...]
    </rdf:Description>
   </prf:component>
   <prf:component>
    <rdf:Description rdf:ID="BrowserUA">
     <prf:BrowserName>Nokia Mobile Internet Client</prf:BrowserName>
     <prf:BrowserVersion>3.0</prf:BrowserVersion>
     [...]
    </rdf:Description>
   </prf:component>
   <prf:component>
    <rdf:Description rdf:ID="WapCharacteristics">
     <prf:WapDeviceClass>C</prf:WapDeviceClass>
     <prf:WapVersion>1.2.1</prf:WapVersion>
     [...]
    </rdf:Description>
    [...]
   </prf:component>
 </rdf:Description>
</rdf:RDF>
```

[7] More information on the Open Mobile Alliance is available at `http://www.openmobilealliance.org`.

[8] The complete UAProf of the Nokia 3650 and many other examples are available at `http://w3development.de/rdf/uaprof_repository`.

MPEG-21 Digital Item Adaptation – Usage Environment Description

The last context-description standard we discuss is the *Usage Environment Description* (UED) tool within the MPEG-21 Part 7 – *Digital Item Adaptation* (DIA) specification. More information on the other tools of MPEG-21 Part 7 and the other parts of MPEG-21 can be found in [52, 53, 61, 62].

The UED tool contains four distinct parts, each containing several subparts. The specification defines the semantics – by normative text – and the syntax – by an XML Schema – of these four parts and their subparts. By defining strict semantics, all UED-compliant applications understand and interpret the information in a similar way. By defining an XML Schema, the context description can be stored and exchanged in a verifiably valid XML syntax. The combination of the two results in a very strict representation of the context. This is an advantage to create automated systems using this information. The disadvantage of such a strict definition is the fact that it is not possible to extend or add new parts to the description to accommodate missing or application domain-specific features.

A detailed overview of the MPEG-21 DIA-UED specification is given in Section A.1 in Appendix A.

2.3.2 Software Toolkit

From the three discussed context description specifications, only the MPEG-21 DIA-UED standard is generically applicable. Indeed, HTTP Headers are only useful for Web page requests and CC/PP is just a framework. The UAProf technology that is based on CC/PP is too specific and only useful for cell phones. It is possible to construct our own CC/PP profile, however this profile is proprietary and can only ensure interoperability if it is supported by the industry.

As the UED tool is a large specification, doubts arise on its usability on constrained devices. We want to evaluate whether or not it is feasible for a memory and processing limited terminal to read, manipulate, and write XML data valid to the MPEG-21 specification. At the same time, we want to facilitate the creation of UED-compliant applications by ensuring that an application does not need to read and write XML data.

Toolkit Contents

To start, we made a thorough analysis of the UED standard using *Unified Modeling Language* (UML) techniques[9]. This resulted in a detailed *Class Model* of the specification (see Section A.2 in Appendix A).

This class model allows us to create a Java-based software toolkit, as illustrated in Figure 2.4, that consists of a *parser*, an *application programming interface* (API), and a *serialization tool*. Our parser is a tool that can read XML files which are valid to the UED specification, i.e., valid to the MPEG-21 DIA-UED XML Schemas. The information in this XML file is poured into an *object tree*. The object tree is an in-memory and hierarchical representation of the UED parts and subparts. The API defines and controls access to this object tree. Typical API operations on a UED (sub)part are read, add, change, and remove. An example of the API operations can be found in Appendix A, Section A.4. It is also the responsibility of the API to block illegal operations on the object tree. For example, it rejects an addition of a second subpart where the specification defines that at most one is allowed. Furthermore, the API also contains navigation operations to move through the object tree, for example, get a specific subpart, and get the parent of the current part. Finally, the serialization tool writes the information in the object tree to an XML data stream that is guaranteed to be valid to the UED XML Schemas, in other words, valid to the MPEG-21 DIA-UED specification.

Toolkit Implementation and Test Set-up

Our goal was to create a software toolkit that can be used on a constrained device; hence we implemented our toolkit such that it works on a cell phone, such as the Nokia 3650. This cell phone runs the *Symbian Operating System* version 6.1 and is able to execute *Java 2 Platform, Micro Edition* (J2ME) applications with the *Mobile Information Device Profile* (MIDP) 1.0 and *Connected Limited Device Configuration* (CLDC) 1.0 configuration[10] – like most recent mobile telephones. It contains 512kb memory heap size for the J2ME applications, which is the minimal size in order to be a J2ME CLDC-compliant device [63].

[9] More information on the Unified Modeling Language is available at `http://www.uml.org`.

[10] More information on Java 2 Platform, Micro Edition is available at `http://java.sun.com/j2me` and in [63].

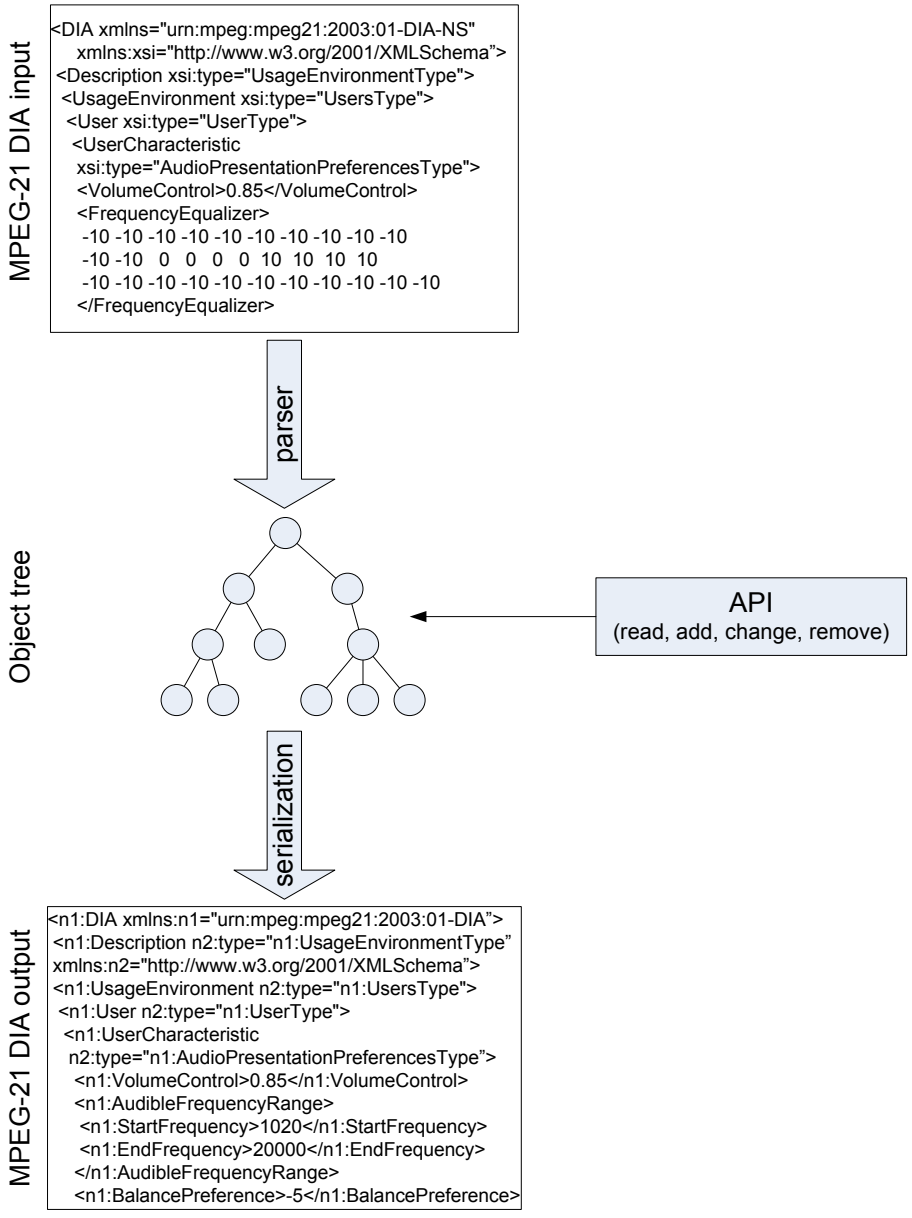


Figure 2.4: MPEG-21 DIA-UED Software Toolkit.

In other words, our toolkit was implemented according to the J2ME MIDP 1.0 / CLDC 1.0 specifications. The size of the toolkit is about 355kb.

Table 2.3: Test set-up for the MPEG-21 DIA-UED Toolkit.

Size software toolkit	363,515 bytes
Input document (i.e., Listing A.1)	8,805 bytes
Object tree manipulations	27 read operations
(excerpt in Listing A.5)	54 change, add, and remove operations
Output document	9,004 bytes

Another important requirement is the feasibility of the toolkit with regard to the execution speed and required memory size. In order to evaluate this, we used following realistic usage scenario (see also Table 2.3). First, a valid DIA-UED document is given as input to the parser (see Listing A.1 in Appendix A). Next, different operations are performed on the data using the API. A part of these manipulations can be found in Listing A.5. Finally, everything is written out using the serialization tool.

In order to determine the time required to execute the steps in this usage scenario, we measured each step independently within a test application that runs on a real Nokia 3650 cell phone. The test was repeated six times without exiting the *Java Virtual Machine* (JVM) and all six tests ran in the same thread. Next, we repeated these six runs of the test ten times but between each execution we restarted the JVM. Hence, the tests were executed sixty times in total.

We also investigated the maximum memory size that is being used while executing the tests. This can be obtained using the *Memory Monitor Extension* tool of the Java *Wireless Toolkit*, which is an integrated part of the J2ME development environment. Although this tool only works within an emulated environments, its results are relevant as the memory architecture of the CLDC-compliant devices is accurately defined.

Results Test Toolkit

Figure 2.5 shows a picture of the output information on the Nokia 3650 cell phone when running the tests.

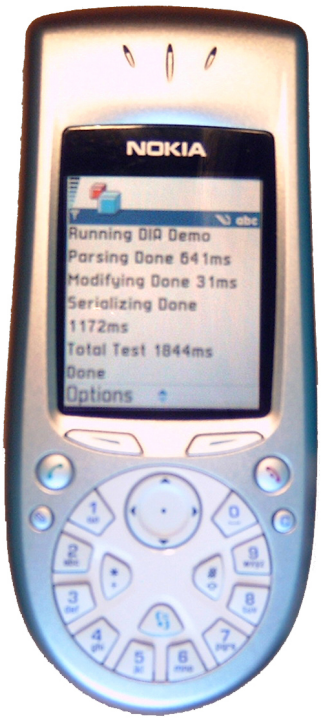


Figure 2.5: Test of the MPEG-21 DIA-UED Software Toolkit on a Nokia 3250 cell phone.

Table 2.4 shows the timing results. We divide the sixty runs of the scenario in two groups. The first group consists of the results for the first time the usage scenario is executed after the JVM was restarted. Hence, it consists of ten values. The average and the standard deviation of these values are shown in Table 2.4 as the *first execution*. The average and the standard deviation of the remaining fifty values are shown in the table as the *subsequent executions*.

The huge difference in execution times between these two groups is undeniable apparent, especially for the parsing operation. During the first execution, all necessary libraries and the input document must be read from the storage card of the cell phone. This storage I/O access is a very expensive operation, which is also proven by the relatively slow serialization time of writing the XML-serialized output to the storage card. However, as parsing and serializing are typically operations performed only once at the start and the end of the application respectively, and

Table 2.4: Execution Times of the MPEG-21 DIA-UED Toolkit Test.

	Parsing (ms)	Processing (ms)	Serialization (ms)	Total (ms)
first execution				
average	10,333	147	1,145	11,625
standard deviation	49	8	7	48
subsequent executions				
average	717	145	1,213	2,074
standard deviation	19	26	40	83

because the API manipulation can be executed relatively fast, these slow operations are not an insurmountable problem to use the toolkit on constrained devices. Also, it can be expected that MPEG-21 DIA-compliant devices load some libraries whilst the device is booting.

The maximum memory used during the execution of the tests was 250,064 bytes. The Memory Monitor Extension tool also allows us to investigate the memory size of a particular object instantiation; the maximum size of the object tree in memory is 21,648 bytes. It is clear that (1) the size of the software toolkit is acceptable even for constrained devices and (2) the heap memory requirement is about 48% of the minimal guaranteed 512kb for J2ME CLDC-compliant devices.

Our software toolkit was submitted to the MPEG consortium and was accepted as the reference software tool for the MPEG-21 DIA-UED specification.

2.4 Related Work

Before concluding this chapter, we briefly give an overview of related research, in particular on content description.

Storing and annotating content in large collections is typically performed by librarians. During the creation of digital libraries of books and images, they have built up a lot of expertise. This expertise could be usable for the creation of digital file-based audio-visual content collections.

The purpose of a digital library – as seen by librarians – is described in [64] as “electronic libraries in which large numbers of geographically

distributed users can access the contents of large and diverse repositories of electronic objects – networked text, images, maps, sounds, videos, catalogues of merchandize, scientific, business and government data sets – they also include hypertext, hypermedia and multimedia compositions." This statement emphasizes that the library community mainly focuses on the disclosure and the exchange of digital objects. This resulted in the creation of the *Metadata Encoding & Transmission Standard* (METS) by the Library of Congress [65]. METS provides a format for encoding metadata used for the management and the exchange of digital objects stored within the library by extending the techniques developed by *the Making of America II* (MOA2) project [66]. However, these standards do not fix the structural, administrative, and technical metadata normatively, while these metadata are essential for audio-visual content description and intended usage in UMA-compliant applications as investigated in this thesis. Furthermore, these standards only refer to available techniques in the pre-digital libraries community for descriptive metadata, such as *Machine-Readable Cataloging* (MARC) records [67] or the *Encoded Archival Description* [68]. Also, these pre-digital documentation techniques are inadequate to fully document digital works.

Better suited standards for the digital libraries are more and more being investigated and used, such as the *General International Standard Archival Description* [69] and the previously discussed DCMES and MPEG-7 standards. In [70] these standards are discussed with regard to their applicability for annotating books and images. The *Open Archive Initiative Protocol for Metadata Harvesting* is one of the first major efforts to address this issue and has selected DCMES to ensure interoperability [71–73]. Unfortunately, the used techniques for creating and maintaining digital libraries of books and images – based on METS and the results of the MOA2 project – "lack adequate provisions for encoding of descriptive metadata, only supports technical metadata for a narrow range of text- and image-based resources, provides no support for audio, video, and other time dependent media, and provides only very minimal internal and external linking facilities" [41]. This implies that solely using these technologies to create audio-visual digital libraries is insufficient. These concerns are addressed by new metadata standards with as main purpose to annotate and manage audio-visual content. Several of them have been discussed in this chapter. However, in contrast to the situation in the digital library community, there is currently a plethora of audio-visual content annotation standards. As such, our evaluation criteria allow one to compare these standards and

select the most appropriate for his intended usage scenario.

2.5 Conclusions and Original Contributions

In this chapter, we investigated various solutions to describe "what" is being consumed and "how" this is being consumed. This information – *metadata* – is needed by an adaptation engine to modify content in such a way that it is consumable for the given context, i.e., the UMA principle.

First, the metadata description tools for the annotation of audio-visual content were studied – the "what." We evaluated the usefulness of the tools that are currently used by the Flemish television broadcasters in order to reuse them. Unfortunately, our studies showed that these tools are proprietary and intended for tape-based content annotation, not for digital file-based audio-visual content.

Luckily, many international content-description standards became available over the last few years. To compare and evaluate these, we have determined four main evaluation criteria. Our criteria can be used to determine the most optimal content-description standard for a particular use case. After evaluating four well-known content-description standards, namely Dublin Core Metadata Element Set, MPEG-7, P/Meta, and the Standard Media Exchange Framework, we concluded that MPEG-7 is most appropriate to annotate audio-visual content in order to realize UMA.

Next, three context-description tools were described – the "how" – namely HTTP Headers, Composite Capability / Preference Profiles, and the MPEG-21 Digital Item Adaptation – Usage Environment Description tools. Unfortunately, no other generically applicable context-description standard is available to date. From the discussion, it became apparent that the MPEG-21 specification was the most useful technology for the intended use in our UMA architecture.

However, as this is a very large specification, we had our doubts on its usability on constrained devices, such as a cell phone. To check our concerns, we created a software toolkit that can be used on the Java 2 Micro Edition platform, i.e., a platform that is more and more available on constrained devices, such as cell phones. Our toolkit is also intended to simplify the development of UED-compliant applications. Hence, it

contains three parts: a parser that reads UED data and converts it to an object tree; an API that allows manipulation of and navigation in that tree; and a serialization tool that writes the object tree in an MPEG-21 DIA-UED compliant XML stream.

Tests of the software toolkit demonstrated its usability in terms of memory requirements and execution speed even on cell phones, although the start-up time was rather long due to the low I/O throughput of the cell phone's storage card.

Our software toolkit was submitted and accepted by the MPEG consortium as the reference software tool for the MPEG-21 Digital Item Adaptation – Usage Environment Description tool.

Our research with regard to content and context description tools resulted in the following papers and contributions to MPEG.

1. Jeroen Bekaert, Dimitri Van De Ville, Boris Rogge, Sam Lerouge, Robbie De Sutter, Emiel De Kooning, and Rik Van de Walle. Metadata-based Access to Multimedia Architectural and Historical Archive Collections. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 22–29, Boston, Massachusetts, USA, July 2002

2. Jeroen Bekaert, Robbie De Sutter, Rik Van de Walle, and Emiel De Kooning. Metadata-based Access to Complex Digital Objects in Multimedia Archival Collections. In *Proceedings of Euromedia 2003*, pages 10–13, Plymouth, United Kingdom, April 2003

3. Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Digital Item Adaptation – Usage Environment Description Tool Parser. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10387, Waikoloa, Hawaii, USA, December 2003

4. Davy De Schrijver, Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. Digital Item Adaptation – Reference Software Tests. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10436, Waikoloa, Hawaii, USA, December 2003

5. Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Evaluation of Usage Environment Description Tools. In *Proceedings of the 2004 International Conference on Internet Computing*, pages 66–72, Las Vegas, Nevada, USA, June 2004

6. Davy De Schrijver, Robbie De Sutter, and Rik Van de Walle. Report on Core Experiment on BSDL extensions. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M12611, Nice, France, October 2005

7. Robbie De Sutter, Stijn Notebaert, Laurence Hauttekeete, and Rik Van de Walle. IPEA: the Digital Archive Use Case. In *Proceedings of the IS&T Archiving 2006*, pages 182–186, Ottawa, Canada, May 2006

8. Robbie De Sutter, Stijn Notebaert, and Rik Van de Walle. Evaluation of Metadata Standards in the Context of Digital Audio-Visual Libraries. *Lecture Notes in Computer Science*, 4172:220–231, September 2006

Chapter 3

Negotiation

3.1 Introduction

Using the content and the context information, a *content adaptation system* can decide how to modify the content in such a way that it becomes usable in the given context. In this chapter, we investigate the possibilities of the location of such an adaptation system in the UMA architecture. Wherever this service is located, it must have the content and context information at its disposal. In other words, the XML-based data must be transmitted over the network to the adaptation system. On top of that, this system must be informed whenever updates to the context occurs, for example, a change in available bandwidth or a significant drop in processing capacity due to the launch of a concurrent application. The entire process of informing the adaptation system about the content and context, including updates thereof, in order to decide upon the ideal version of the content is called *negotiation*. Negotiation is an important part in any UMA-compliant architecture as it allows the adaptation engine to select or to construct a suitable version of the content in such a way that the content can be consumed in the particular context.

In this chapter, we first discuss the different location strategies for the adaptation system in the UMA architecture. Next, we investigate solutions to send XML-based data over a network and introduce our concept of *time-varying metadata*. Finally, we demonstrate the impact of a factor that can not be underestimated during the negotiation, namely the overhead due to the verboseness of the XML-based data.

3.2 Content Adaptation

Figure 3.1 illustrates the concept of a *content adaptation system*. Using the content and context information, a *content adaptation decision engine* determines the conditions and requirements, also called the *adaptation rules*. Together with the actual content, the rules are given as input to the *content adaptation engine*. This engine actually performs the content adaptation, which results in *optimized content*, i.e., content that is usable in the given context.

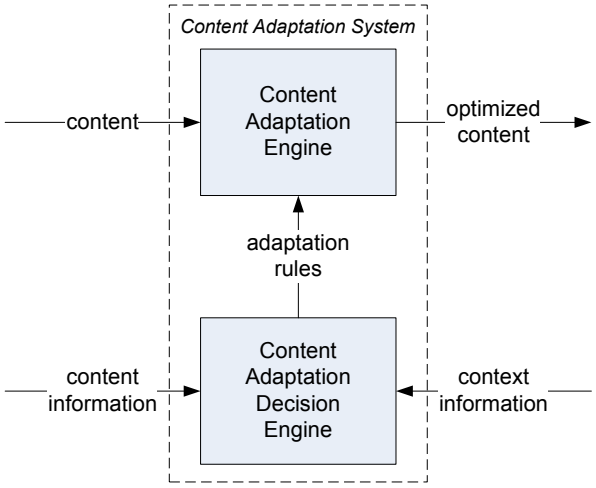


Figure 3.1: A *content adaptation system*: the *content adaptation decision engine* instructs the *content adaptation engine* how to modify the original content into optimized content. This decision is based on information about the content and context.

In [26,74,75] a technique is proposed so the decision about the adaptation rules is dealt with as a multi-criteria optimization problem, which can be solved by the mathematical Pareto theory. In this thesis, we take abstraction of the inner workings of the content adaptation decision engine. However, we will discuss the different possibilities for audio-visual content adaptation in Chapter 5 "Video Scalability."

In this chapter, we investigate *where* the content adaptation engine and the content adaptation decision engine are located in the UMA architecture and *how* the content and context information are negotiated with the latter. Again, we assume our use case of an end user who wants to consume an optimized audio-visual stream from a content provider over

a network on his device. Hence, we will also assume that the audio-visual stream is encoded by a *conventional*, non-distributed approach. Indeed, in the conventional approach, the video *encoder* is computationally more complex than the video *decoder*, whereas in the *distributed video coding* model the opposite is true [76]. Because we focus on a Video-on-Demand and/or broadcasting scenario, the non-distributed approach is most suitable.

3.2.1 Location of the Content Adaptation Engine

There are four possibilities to locate the content adaptation engine:

- *Content server*: the content adaptation is performed by the content provider, hence close to the original content source. This has as advantage that the content provider has, to some extent, control over the adaptation process. For example, the streaming of the audio-visual content can be aborted if the optimized content does not achieve the minimal required quality level imposed by the content provider. On the other hand, performing content adaptation on the content server increases the complexity and workload for the content provider.

- *End-user device*: the adaptation is performed on the terminal itself. This has as advantage that the same content can be sent to all requesting devices, as such it migrates the responsibility for optimized content delivery to the receiving device. A big disadvantage is the fact that bandwidth is wasted. Indeed, only a part of the received data is actually consumed on the client device. In addition, this solution is not always feasible, especially in constrained environments, because the network or the end-user device cannot process the data in real-time.

- *Broker service*: a dedicated service performs the content adaptation. The advantage is that a separate system is responsible and hence optimized to perform one task, namely adaptation. A disadvantage is an increased delay as the content must pass through an additional system.

- *Network*: content adaptation is performed by nodes in the network, e.g., a gateway, a router, etc. An advantage is that the content adaptation can be divided in very small steps and spread

out over many nodes. As such, the delay due to the adaptation process is kept as low as possible. A major disadvantage is the increased complexity for the network nodes as these have to perform additional tasks they are not primarily designed for.

It is also possible to combine two or more locations such that the content is adapted several times, for example reducing the number of frames to comply with the available bandwidth in the network and scaling the resolution down on the end-user device to the characteristics of the device. Although this spreads the complexity of the adaptation over multiple devices, the overhead and the additional delay is an issue. Hence, in the remainder of the section, we assume that the content adaptation occurs at one location.

Locating the adaptation engine at the end-user device or in the network increases the complexity and overhead too much, making these two locations not feasible. The broker service is a good solution as long as the additional delay is not a blocking factor. On the other hand, if the additional complexity and workload can be kept to a minimum, locating the content adaptation at the content server also provides a good solution.

Usually, relatively simple and straightforward content adaptation schemes, for example changing the frame rate, can be handled by the content server mostly by tweaking parameters of the audio-visual streaming server or truncating the bit stream. More advanced adaptation schemes, such as changing the resolution or the bit rate of a non-scalable video stream, are typically performed by a dedicated service, like a broker service. More information about video scalability will be given in Chapter 5.

In the remainder of this chapter, we will assume a straightforward content adaptation scheme, hence we locate the content adaptation engine at the content provider.

3.2.2 Location of the Content Adaptation Decision Engine

The same four locations are feasible for the content adaptation decision engine, but the location of both engines does not necessarily need to coincide. For each possible location, we illustrate the flow of the data messages that must be exchanged in order to stream optimized video content to an end-user device. Wherever the service is located, it must

have the content description information and the context description information from the end-user device, the network, and the content server at its disposal. The adaptation rules are the result of the service.

- *Content server (Figure 3.2)*: the decision is taken by the content server. To do so, the engine only requires the context information from the end-user device and the network as it has direct access to the information about its own context and the content.
- *End-user device (Figure 3.3)*: the content information and the context information about the content server and the network are transmitted to the end-user device. Together with the known information about the end user and the device itself, the decision-taking engine can determine the adaptation rules.
- *Broker service (Figure 3.4)*: the decision engine is located at a dedicated service. It receives the required information about the content and the context and based on this data the adaptation rules are determined.
- *Network (Figure 3.5)*: it is possible to decide upon the adaptation rules in the network by informing one or more network nodes about the content and the end user, the end-user device, and the content server context information.

All possible locations for the content adaptation decision engine are feasible. Nevertheless, locating this engine at a dedicated broker service provides scalability advantages. Indeed, if the requested audio-visual content is available at different content providers or different (geographically distributed) content servers, the broker can send the request to the most suitable server (e.g., in terms of workload or nearest to the end-user device).

In the remainder of this chapter, we assume that the content adaptation decision engine is located at a broker service. However, our explanation hereafter is also valid for all other locations.

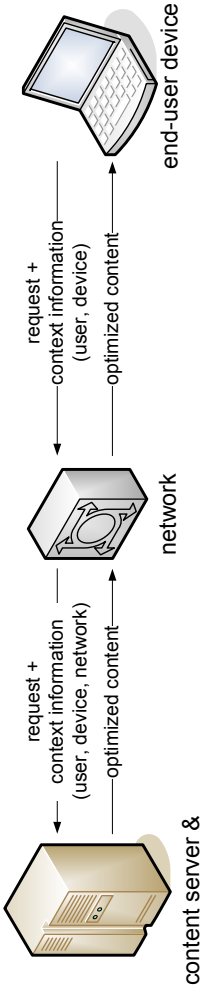


Figure 3.2: The content server decides upon content adaptation rules.

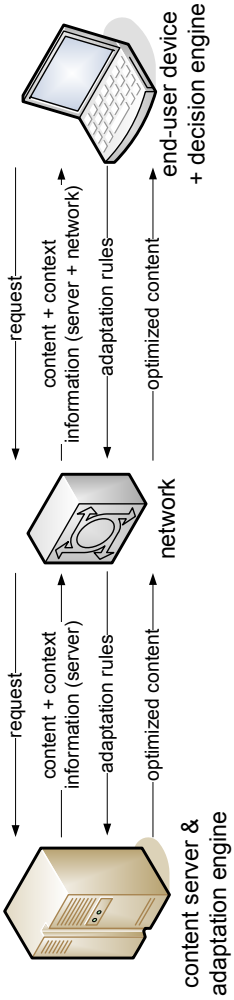


Figure 3.3: The end-user device decides upon content adaptation rules.

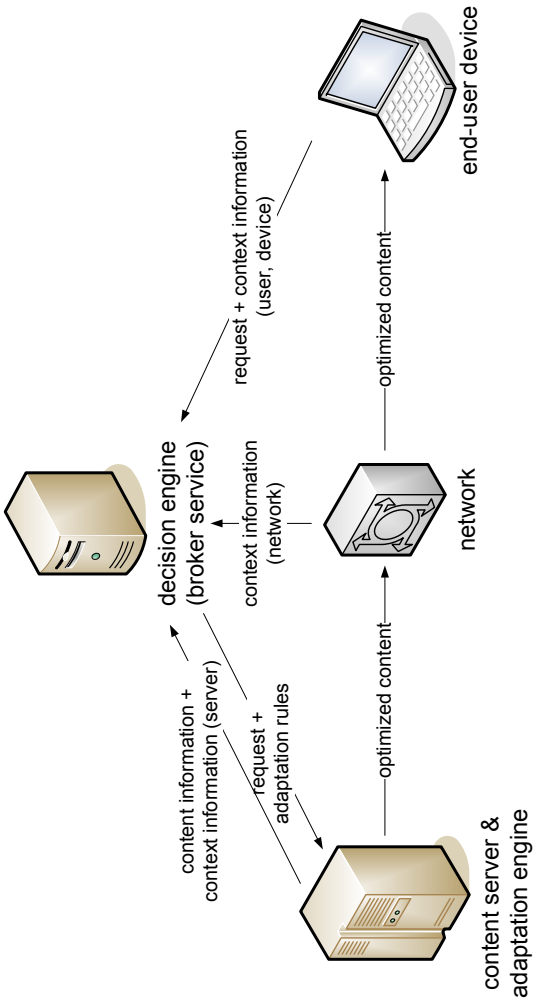


Figure 3.4: The broker decides upon content adaptation rules.

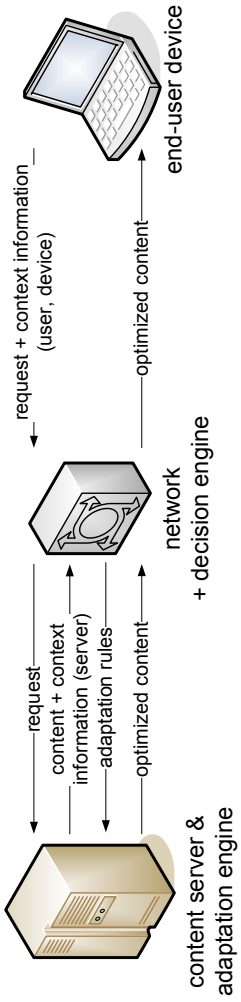


Figure 3.5: The network decides upon content adaptation rules.

3.3 Invoking the Content Adaptation Decision

Wherever the content adaptation decision engine is located in the UMA architecture, information about the content, context, or both must be negotiated. This implies transmitting the information over the network to the engine.

Creating a content adaptation decision engine as an independent broker service means that all content and context related information must be transmitted over the network. Indeed, the broker service needs information about the content and context from the content provider, network context information from the network, and end user and end-user device context information from the terminal. This is illustrated in Figure 3.4.

As explained in the previous chapter, the content and context information is structured using XML. Hence, in order to transmit the content and context information to the content adaptation decision engine, we have to investigate how to transmit XML-based data over a network.

However, we can look at this issue from a different perspective. The decision-taking engine takes as input various parameters (namely the content and context information) and executes its algorithms using these parameters to determine the adaptation rules. In other words, we invoke a *service* that resides on a remote server and that will perform the necessary calculations in order to decide about the adaptation rules. In the remainder of this section, we explain the two most relevant techniques that are designed to invoke these so called *web services*.

3.3.1 XML-RPC

The *Extensible Markup Language – Remote Procedure Call* (XML-RPC[1]) was the first XML-based technique created to invoke web services. It was designed as a lightweight and easy solution in contrast to previous non-XML-based techniques, such as CORBA[2] and DCOM[3]. Indeed, XML-RPC simplifies interoperability and is easier to implement. Furthermore, it uses the HTTP protocol as transport protocol. As a re-

[1]More information on XML-RPC is available at `http://www.xmlrpc.com`.

[2]More information on the *Common Object Request Broker Architecture* (CORBA) technology is available at `http://www.corba.org`.

[3]More information on Microsoft's *Distributed Component Object Model* (DCOM) technology is available at `http://msdn.microsoft.com/library/en-us/dndcom/html/msdn_dcomtec.asp`.

sult, XML-RPC clients are having fewer problems to communicate with servers than the older transport protocols like CORBA and DCOM.

An example of an XML-RPC message can be found in Listing 3.1. This example illustrates the simplicity of XML-RPC. The lines 1 to 5 represent the classical HTTP-request Header. The `content-type` on line 4 specifies that the content is XML-based data (`text/xml`). The XML-RPC request message itself is constructed as follows:

- The root node is called `methodCall` (line 8).
- The first child element of the root node is named `methodName` and contains the name of the method that is called (line 9). It is up to the receiver to interpret this name and execute the appropriate service.
- The second child element of the root node (`params`) is optional and groups the parameters (stored separately in `param` elements) for the service (lines 10 to 14). These `param` elements must appear in the same order as in the web service's signature. The XML-RPC standard specifies six data types in which the parameter value can be stored: a four-byte signed integer, a boolean, a string, a double-precision signed floating point number, an ISO formatted date and time, and a base64-encoded binary value. On line 12 of the example, a four-byte signed integer is used.

Listing 3.1: XML-RPC message (including HTTP Header fields).

```
1  POST /demo HTTP/1.0
2  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
3  Host: www.ugent.be
4  Content-Type: text/xml; charset=utf-8
5  Content-length: 219
6
7  <?xml version="1.0"?>
8  <methodCall>
9   <methodName>examples.setAvailableBandwidth</methodName>
10  <params>
11    <param>
12      <value><int>9600</int></value>
13    </param>
14  </params>
15 </methodCall>
```

The creation of an XML-RPC message can easily be accomplished by various tools. The most well-known tool is Microsoft's *XMLHTTP* API of which alternative implementations exist that run on non-Microsoft-based platforms, such as the W3C's *XMLHttpRequest object*[4]. This component is also used in the upcoming *Asynchronous JavaScript and XML* (AJAX) technology[5]. Although AJAX is intended to create "snappier" (Web-based) user interfaces – for example, it is used in Google's GMail e-mail service[6] – it can also be used to invoke remote procedures, mostly because it allows asynchronous (XML-based) data exchange.

3.3.2 SOAP

The simplicity of the XML-RPC technology and, in particular, the restricted set of possible data types for the parameter values resulted in the development of a more advanced technology, namely the *Simple Object Access Protocol* (SOAP) [77–79]. Just as the XML-RPC technique, SOAP uses HTTP as transport protocol; bindings to other protocols, such as the *Simple Mail Transport Protocol* (SMTP) are also supported. Theoretically, a *SOAP message* is a one-way transmission between *SOAP nodes*, from a *SOAP sender* to a *SOAP receiver*, optionally via additional *SOAP intermediaries*. However, in practice, SOAP messages are usually bi-directional and are often complemented with additional advanced techniques (such as routing and multi-casting).

In general, a SOAP message consists of two parts encapsulated in a *SOAP Envelope* (see also Listing 3.2):

- an optional *SOAP Header* (lines 10 to 15): this block contains additional information about the payload and is, as such, no part of the payload. The SOAP Header is an extension mechanism for SOAP applications. More information about the advanced standardized features of the SOAP Header are discussed hereafter.

- a mandatory *SOAP Body* (line 16 to 30): the actual payload of the message intended for the SOAP receiver. The payload must be structured in the XML format.

[4]More information the XMLHttpRequest object is available at `http://www.w3.org/TR/XMLHttpRequest`.

[5]More information on AJAX is available at `http://www.ajaxmatters.com`.

[6]Google's GMail e-mail service is available at `http://www.gmail.com`.

Listing 3.2: SOAP Message example (including HTTP Header fields).

```
1  POST /demo HTTP/1.0
2  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
3  Host: www.ugent.be
4  Content-Type: text/xml; charset=utf-8
5  Content-Length: 1542
6
7  <?xml version="1.0" ?>
8  <soap:Envelope
9   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
10  <soap:Header>
11   <U:type xmlns:U="http://www.ugent.be"
12      soap:role="http://www.w3.org/2003/05/soap-envelope/role
            /next"
13      soap:relay="true"
14      soap:mustUnderstand="true">X</U:type>
15  </soap:Header>
16  <soap:Body>
17   <DIA
18    xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
19    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
20    <Description xsi:type="UsageEnvironmentType">
21     <UsageEnvironmentProperty xsi:type="TerminalsType">
22      <Terminal>
23        <TerminalCapability xsi:type="DisplaysType">
24        <!-- Full Listing A.2 can be found in Appendix A. -->
25        </TerminalCapability>
26       </Terminal>
27      </UsageEnvironmentProperty>
28    </Description>
29   </DIA>
30  </soap:Body>
31 </soap:Envelope>
```

The listing above illustrates the advanced standardized features of SOAP messaging with regard to the SOAP Header [78]. Each SOAP Header child element can contain three attributes with special connotations:

- `role`: the `role` attribute specifies the SOAP node that must process this SOAP Header child element (hereafter referred to as SOAP Header block). There are three default roles defined in the SOAP specification[7]:

[7] We list the (default) abbreviations of the standardized roles. Each `role` value must be prefixed with `http://www.w3.org/2003/05/soap-envelope/role/` as illustrated on line 12 in Listing 3.2.

  - `next`: all SOAP intermediaries and the SOAP receiver must process this SOAP Header block.
  - `none`: no SOAP node (intermediaries and the receiver) can process this SOAP Header block. However, it may be necessary to investigate its value if a SOAP node is instructed to do so (for example, another SOAP Header block).
  - `ultimateReceiver`: only the SOAP receiver must process this SOAP Header block.

  It is possible to use a proprietary value as role. If a SOAP node fits the role, then it must process the SOAP Header block. It is not defined by the SOAP specification how this check must be performed. By default, this attribute value is `ultimateReceiver`.

- `relay`: if this attribute contains the boolean value `true`, then the SOAP Header block must be present when the SOAP message is forwarded. This overrules the default work method defined in the SOAP specification as this states that a SOAP Header block must be removed after it has been processed, even if this is done by a SOAP intermediary node. By default, this value is `false`.

- `mustUnderstand`: this boolean attribute determines that a processing SOAP node (as determined by the `role` attribute) must "understand" the information in the current SOAP Header block. If this is required (a value `true`), but the node does not "understand" the information, an error message must be returned to the SOAP sender. According to the SOAP specification "understanding a header means that the node must be prepared to do whatever is described in the specification of that block" [77]. By default, this value is `false`.

Without any of the enumerated attributes – thus, the default, – a SOAP Header block is optionally processed only by the SOAP receiver. The SOAP Header block in Listing 3.2 must be processed and understood by all SOAP intermediaries without removing it and by the SOAP receiver.

The flexibility of the SOAP specification makes it possible to create more advanced *Web Service Infrastructures*. The main idea of a Web Service Infrastructure is to develop a library of remote (Internet-based) procedures (i.e., *services*) that can easily be found by any client. Once found, these services can be called using SOAP messages.

A *service broker* manages the library of services and publishes them using the *Universal Description, Discovery, and Integration* (UDDI) specification[8]. This industry-driven open specification defines a registry service that service providers can use to expose their services. The service consumers can use this to discover the services, hence it can be seen as a telephone directory for Web Services.

To complete the infrastructure, the W3C is developing the *Web Services Description Language* (WSDL)[9]. It specifies how to exchange information about the service. This XML-based description contains information about the goal of the service (for example, a small description of the service), the name of the service, the allowed network bindings (for example, HTTP and SMTP), the name of the parameters and their data types, the expected return value data type, and so on. Finally, the SOAP message itself is extended. The concept of *SOAP Attachments*[10], similar to e-mail attachments, and *digital signatures for SOAP messages*[11] are two examples of such extensions.

3.4 Exchanging XML-based Information

3.4.1 Work Method

The technologies described in the previous section make it possible to invoke web services. Because we regard the content adaptation decision engine as a web service, we can use one of the two described techniques in our UMA architecture.

The decision engine is steered by three parties, each party sends specific information as input to the service (see also Figure 3.6):

1. The content provider (i.e., the *content server + adaptation engine*) sends as parameter value the XML-based content information data

[8] More information on UDDI is available at `http://www.uddi.org`.

[9] WSDL is not yet an endorsed standard as the first versions are a W3C Note. Currently, W3C is developing a new version of WSDL that will become a W3C recommendation. More information on WSDL is available at `http://www.w3.org/2002/ws/desc`.

[10] More information on SOAP Attachments is available at `http://www.w3.org/TR/SOAP-attachments`.

[11] More information on digital signatures for SOAP messages is available at `http://www.w3.org/TR/SOAP-dsig`.

valid to the MPEG-7 specification and the MPEG-21 DIA-UED based context information.

2. The *end-user device* sends as parameter value the XML-based context information data about the end user and the end-user device valid to the MPEG-21 DIA-UED specification.

3. The *network* sends as parameter value the XML-based context information data about the network valid to the MPEG-21 DIA-UED specification.

If the RPC technology does not support XML-based data as a data type – i.e., XML-RPC – the XML-based data can still be transmitted as a plain-text "string" data type.

Next, the content adaptation decision engine processes the received information, for example using the MPEG-21 DIA-UED toolkit as discussed in Section 2.3.2. After deciding how the content is adapted, the content adaptation decision engine invokes on its turn the content adaptation engine. The latter can also be seen as invoking a web service. The adaptation engine exposes the adaptation methods it supports, for example, stream at half the frame rate.

3.4.2 Time-Varying Metadata

So far, we discussed the traditional Universal Multimedia Access concept: content is optimized to suit the context. In the following, we refine this concept by taking a *changing* context into account. As such, we re-optimize the content if needed. For example, an end user is consuming content on a portable computer connected to a wireless network. When he connects this computer to a fixed local network, the available bandwidth will (usually) increase. The content adaptation decision engine must become aware of this change as it might alter the original decision on how to optimize the content. In this example, it might become possible to send more frames per second thanks to the expanded network capacity.

We define this kind of metadata as *time-varying metadata*, namely metadata which value is likely to change on relatively short term, for example during the consumption of an audio-visual stream.

The definition and concept of time-varying metadata was introduced by us in [8].

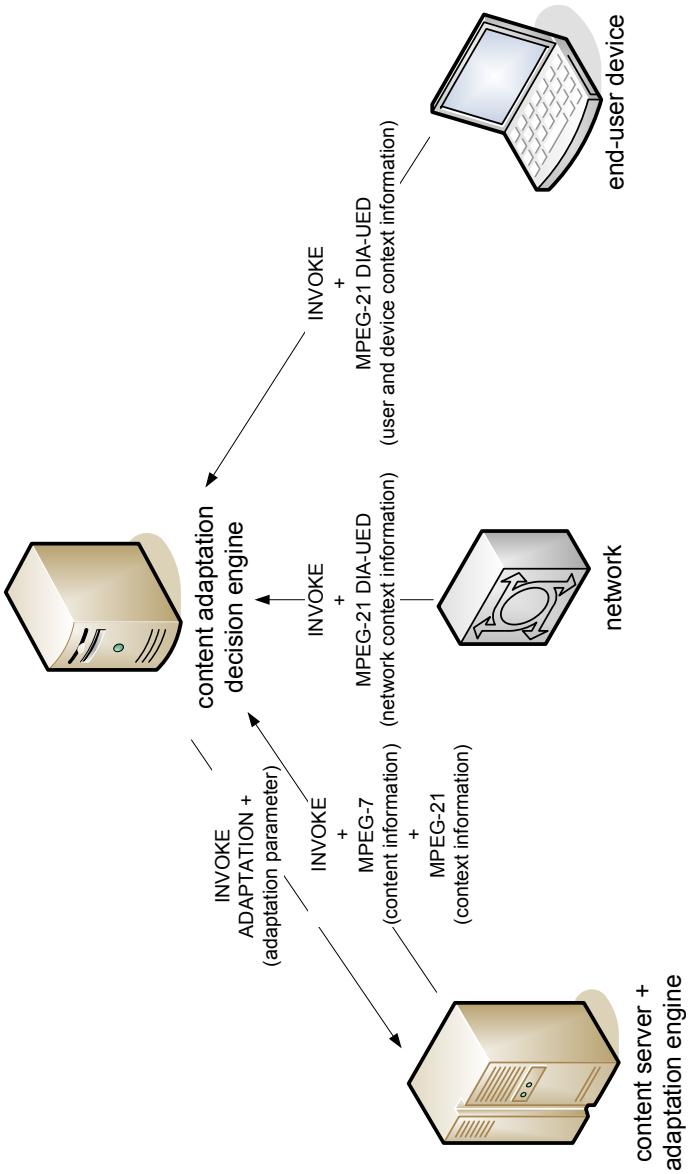


Figure 3.6: Invoking web services on the content adaptation decision engine and on the content adaptation engine.

In our use case, time-varying metadata is information about the context that can change during the consumption of the content. This is in contrast to the information about the content that is fixed – at least during the consumption of the content.

If the time-varying metadata changes (i.e., the context is changed), the client or the network must re-send the context information to the broker. As such, this new context can result in a change in the adaptation rules such that a re-optimization of the content can occur. It is imperative to create a smooth transition. In other words, we do not accept that the user must restart the session or that the re-optimization involves a delay during which the user does not receive audio-visual content. As such, our content adaptation engine must work under real-time constrains.

3.4.3 Problems and Concerns

Sending complete and valid XML-based data over a network introduces overhead. Not only is the plain-text serialization of the XML-based data verbose, its tree-like structure generates overhead as well. In addition, support for the time-varying metadata involves more XML-based data messages to be transmitted. Indeed, each time a relevant change to the context occurs, the context information (being the network information, the end user and end-user device information, or the content server information) must be transmitted. For example, if the available bandwidth changes, the network must send a message similar to the one depicted in Listing A.2 in Appendix A.

It is possible to enhance the content adaptation decision engine by exposing web services particularly usable to exchange the time-varying metadata values. For example, the engine could expose a "setAvailableBandwidth" method that accepts as parameter an integer representing the current available bandwidth in bytes per second. As such, the network does not need to send a complete and valid UED message, but only this value, hence reducing the overhead. For example, comparing this solution to the complete UED-valid message as shown in Listing A.2 means a reduction from about 650 bytes to 4 bytes[12].

[12]We assume that all characters of the XML-based data message string can be represented by 1 byte, although *8-bit Unicode Transformation Format* (UTF-8) characters can take up to 4 bytes in particular cases. An integer data type is represented in 4 bytes by most programming languages.

The downside of this solution is an increased complexity for the content adaptation decision engine as well as for the clients as more web services must be implemented and used correctly. Hence, such a system forfeits its flexibility and future compatibility. Indeed, an update on the structure of the information – thus, the XML Schemas – requires an update of the interfaces, for example, a change in data type of a parameter. If an end-user device is created according to an older version, it sends the wrong kind of data or uses the wrong kind of web service. However, if XML-based data is transmitted, the receiver can determine the versions of the XML Schemas that are used from, for example, the specified namespace in the XML data.

In the following chapter, we solve the overhead of XML-based data negotiation using alternative XML serialization formats instead of the traditional plain-text notation.

3.5 Conclusions and Original Contributions

In this chapter, we discussed two important parts in our UMA architecture, namely the negotiation and the content adaptation. The content adaptation must ensure that the audio-visual content is optimized so it becomes usable on a particular device, for a particular end user, and for a particular network, hence the context. As such the content and context information is negotiated in order to optimize the audio-visual content.

We divided the content adaptation in two separate parts: a content adaptation decision engine and a content adaptation engine. The former determines, based on the information about the context, how to modify the audio-visual content, the latter performs the actual adaptation. Next, we investigated the possible locations to place both engines in the UMA architecture. It is not necessary that these coincide. Indeed, it was decided to locate the content adaptation engine near the content, hence at the content provider, and to place the content adaptation decision engine as a separate part in the architecture.

Anywhere the content adaptation decision engine is located, it must have the information about the content and the context at its disposal. Hence, the XML-based data must be negotiated and transmitted over the network. We took abstraction from the decision-taking engine by regarding it as a web service. Indeed, the engine takes input parameters

and processes them, resulting in the rules for the content adaptation engine.

Invoking a web server is typically done by a using the XML-RPC or the SOAP technologies. We further discussed how this web service invocation concept fits in our UMA architecture.

Next, we introduced our concept of time-varying metadata, which is our original contribution to the UMA framework. The main idea is to dynamically optimize the content to a changing context by re-negotiating the context information. As such, we are able to handle alterations in the environment automatically and re-optimize the content on-the-fly.

To conclude this chapter, we discussed an issue that occurs when negotiating XML-based data, namely overhead. Indeed, XML-based data are verbose due to the usage of a plain-text serialization and structural overhead. This issue could be solved by enhancing and extending the web services of the content adaptation decision engine, however this would imply more complexity and a loss of flexibility. As such, we will investigate other solutions to address the overhead issue in the next chapter.

The research that has resulted in this chapter of this thesis is also discussed in the following publications.

1. Robbie De Sutter, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Adapting Mobile Multimedia Applications to Changing End-User Preferences. In *Proceedings of Euromedia 2002*, pages 180–182, Modena, Italy, April 2002

2. Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Dynamic Adaptation of Multimedia Data for Mobile Applications. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 240–248, Boston, Massachusetts, USA, July 2002

3. Sam Lerouge, Boris Rogge, Robbie De Sutter, Jeroen Bekaert, Dimitri Van De Ville, and Rik Van de Walle. A Generic Mapping Mechanism between Content Description Metadata and User Environments. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 12–21, Boston, Massachusetts, USA, July 2002

4. Boris Rogge, Robbie De Sutter, Jeroen Bekaert, and Rik Van de Walle. An Analysis of Multimedia Formats for Content Descrip-

tion. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 1–11, Boston, Massachusetts, USA, July 2002

5. Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, and Rik Van de Walle. Dynamic Adaptation of Streaming MPEG-4 Video for Mobile Applications. In *Proceedings of Euromedia 2003*, pages 185–190, Plymouth, United Kingdom, April 2003

6. Robbie De Sutter, Sam Lerouge, Wesley De Neve, Peter Lambert, and Rik Van de Walle. Advanced Mobile Multimedia Applications: using MPEG-21 and Time-Dependent Metadata. In *Proceedings of SPIE/ITCom Multimedia Systems and Applications VI*, volume 5241, pages 147–156, Orlando, Florida, USA, September 2003

7. Sam Lerouge, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Fully Scalable Video Coding in Multicast Applications. In *Proceedings of SPIE/Electronic Imaging 2004*, volume 5308, pages 555–564, San Jose, California, USA, January 2004

8. Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Personalizing Quality Aspects in Scalable Video Coding. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, Amsterdam, The Netherlands, July 2005. Published on CD-ROM

Chapter 4

Alternative XML Serializations

4.1 Introduction

In Chapter 2, it is demonstrated that XML is used as the preferred language to represent information about the content and the context. XML has the advantage of being well established and widespread, however its main disadvantage is its verbose characteristic due to the fact that XML data are serialized as plain text. This verboseness results in overhead and should not be underestimated when it is used to exchange information over slow or expensive networks as illustrated in Chapter 3.

This chapter explains our solution to the verboseness issue of XML using *alternative* – i.e., non-textual – XML serialization formats and creating an update functionality for the XML data. First, we investigate the tools that are currently available to process XML-based data. We study the typical features of these tools and classify them into different models. Second, we introduce three techniques that can be used as an alternative for the classic textual notation of XML data, namely ZIP compression, *Abstract Syntax Notation One* (ASN.1), and *Binary MPEG Format for XML* (BiM).

Hereafter, we create our *serialization-agnostic parser*, i.e., a parser that can handle XML data serialized in plain text and in the three discussed alternatives. On top of that, our parser has to support *update functionality*. This requirement ensures optimal handling of time-varying

metadata as discussed in previous chapter. Both requirements must be accomplished without introducing additional complexity for the application or the application developer.

Finally, we evaluate the usefulness of the alternative serialization formats and the XML update functionality using our developed parser in two real-life situations. The first application handles UED-based context information; the second is based on the very popular RSS[1] application.

It must be noted that by using a non-textual format for the serialization of XML data, one of the key characteristics of XML is lost, namely the (human) readability. We do not want to minimize the advantage of this property. Nevertheless, in some cases, we do think that the benefits of a compact representation of XML data are more important than its readability, in particular to address the bandwidth concerns as expressed in the previous chapter. Because the developed parser shields the application and the application developer from any technical details about the non-textual serialized XML data, we believe that using these alternative serialization formats do not create an increased complexity with regard to the usage of an alternative serialization format.

4.2 Parsing XML Data

This section provides a detailed study on existing techniques to handle XML-based data by means of XML parsers. These parsers are catalogued into five XML parser models.

4.2.1 Terminology

In order to understand the remainder of this chapter, it is necessary to define the terminology.

Definition 4.1. *A* parser *is a (software) tool that analyzes and organizes formal language statements into a usable form for a given purpose.*

[1]The abbreviation RSS is sometimes also explained as *Rich Site Summary* or *RDF Site Summary* and reflects to a particular version of the specification. The former reflects to version 0.91, the latter to the versions 0.9 and 1.0. In this thesis, we use version 2.0, which is called the *Really Simple Syndication*. More information on the RSS 2.0 specification is available at `http://www.rssboard.org/rss-specification`.

Definition 4.2. *A* parser model *is an abstract concept that specifies the overall parsing principles that a compliant parser must obey.*

Definition 4.3. XML markup *is any of the following XML structures (as defined in [44]): "a start tag, an end tag, an empty-element tag, an entity reference, a character reference, a comment, a CDATA section delimiter, a document type declaration, a processing instruction, an XML declaration, a text declaration, and any white space that is at the top level of the document entity (that is, outside the document element and not inside any other markup)."*

Definition 4.4. XML character data *is all text that is not XML markup. It is usually the textual content of XML elements.*

Definition 4.5. *An* XML token *(or in short a* token*) is XML markup or XML character data.*

Definition 4.6. *An* XML data stream *is a well-formed sequence of XML tokens.*

Definition 4.7. *An* XML parser *is a software tool that (1) handles the reading of an XML data stream, (2) divides the stream into XML tokens, and (3) makes the XML tokens available according to the rules of the parser model.*

4.2.2 Common XML Parser Functionalities

Currently, there are many different implementations of XML parsers available, intended for various programming languages, platforms, types of applications, and so on. Each parser is implemented according to a specific parsing model. Nevertheless, the base functionality is the same, namely processing *XML data streams*[2]. As such, they are constructed around some common basic concepts. In this part of the section, we give the results of our survey of these concepts. By combining (parts of) these concepts in certain ways, different XML parser models are created with different characteristics.

[2] An *XML data stream* is a flow of (XML-based) data that originates from a locally stored XML file, XML content that is being downloaded from a remote location, or any other source that produces XML-based data.

Bootstrap

Bootstrapping is the entry point of a parsing process. To allow different implementations of the same parser model, an interface offering the bootstrap operations must be defined. The minimal operations are:

- *Load:* resolves and subsequently opens the physical source of the XML data stream, such as a file or an HTTP connection.
- *Prepare:* detects the encoding format of the XML data stream. This can be accomplished using the information in the XML header, MIME-type information, or using a character encoding detection algorithm as suggested in Appendix F of [44].
- *Initialize:* sets up the necessary internal structures such that navigation and token operations, as discussed hereafter, can be executed.

Specific implementations of a parser model can offer additional features, such as, enable XML validation, perform normalization, and support for XML entity replacements.

Navigation

The term *navigation* means the ability to move through the XML data stream to reach a specific token:

- *Simple forward navigation:* every parser model must offer simple forward navigation, either explicit or implicit. Simple forward navigation allows the application to instruct the parser to move through the XML data one token at a time. This basic operation is usually extended – and sometimes concealed – by more advanced forward navigation methods such as: go to the next (start) tag, go to the first child element, go to the last sibling, and similar operations.
- *Simple backward navigation:* the basic operation is to move back one XML token. It is usually extended by more advanced backward navigation methods. Typical examples are: move to the parent node, move to the previous sibling, and move to the root node. Simple backward navigation is offered by some parser models.

- *Random navigation:* this is the most flexible way for navigating through the XML data. It allows the application to command the parser to "jump" directly to certain parts in the XML data stream. For example, the parser can: jump to an element with a specific ID attribute value, jump to a token with a certain fully qualified name, and address a token by an XPath [80] expression. It depends on the parser model if this form of navigation is available.

Token Operations

By applying a navigation operation, the application can select a single token in the XML data stream. We call this token the *current token*. The next step is to act upon the current token by consuming or manipulating it:

- *Token consumption:* all parser models allow token consumption, also called *read* functionality. In this context, *read* means to investigate the current token and harvest its value. The main operation is the retrieval of the token type, i.e., it identifies the current token as an element (a start tag or an end tag), an attribute, text data, a CDATA section, a comment tag, a processing instruction, or ignorable white space. Each token also returns its value. Furthermore, the start tag, the end tag, and the attribute tokens also make their name and namespace information available. Advanced parsers can provide additional information, such as the number of attributes and the number of children in case of a start tag.

- *Token manipulation:* some parser models allow the manipulation, also called *write* functionality, of the XML data stream by adding new tokens or by deleting the current token. Changing the current token can be seen as a delete operation followed by an add operation. The add operation has multiple provisions. For example, it is not allowed to add a processing instruction to an attribute token. The delete operation on an element token must be seen as the removal of the current element and all child elements, if any. Note that this operation does not imply marshalling of the XML data, which is strictly speaking not the responsibility of an XML parser.

Auxiliary Operations

The auxiliary operations are not necessary for the parsing of an XML data stream and, as such, are no part of any parser model. However, they are provided by most parser implementations as an aid for the application, such as localization of the current token in the physical stream, token comparison, namespace prefix lookup, duplication of a token, and so on.

4.2.3 Survey of XML Parser Models

If an application processes an XML data stream, it accesses this data indirectly using one of many available XML parsers. We studied these available parsers and identified five distinct parser models. This section describes each model in detail, starting from the oldest model that is developed by the W3C, namely the *Tree Model*. The other models are the *Push Model*, *Pull Model*, the *Cursor Model*, and the *Mapping Model*. For each model, we list an example of a parser and the typical model characteristics. This information is also summarized in Table 4.1.

Table 4.1: Comparing XML Parser Models.

	Tree Model	Push Model	Pull Model	Cursor Model	Mapping Model
Navigation					
forward	yes	yes	yes	yes	yes
backward	yes	no	no	yes	yes
random	yes	no	no	yes	yes
Token operations					
consumption	yes	yes	yes	yes	yes
manipulation	yes	no	no	optional	yes
Processing speed					
parsing	slow	fast	fast	fast–slow	slow
consumption	fast	medium	medium	slow–fast	fast
manipulation	fast	n/a	n/a	fast	fast
Memory req.	high	low	low	low–high	medium

The Tree Model

This is the original and oldest model that exploits the tree structure characteristic of an XML document. The XML tree is reconstructed in memory in such a way that it reflects the XML data model [81]. The parser grants the applications access to the in-memory tree and offers navigation throughout the tree.

The characteristics of the Tree Model are:

- During the initialization step of the bootstrap, an in-memory tree of the XML data is constructed.
- All navigation methods are available.
- All token operations (token consumption and token manipulation) are available.
- The application is in control of the parser. In other words, the application invokes the navigation methods and the token operations.
- Bootstrapping is slow because the complete XML data stream must be processed in order to build the in-memory tree structure. In addition, the XML data stream must be completely available before the bootstrap method can create the tree, e.g., the data must be entirely downloaded from an HTTP connection. Hence, the tree model can not be used in streaming scenarios. After the bootstrap method, the navigation and token operations are fast because these operations occur on the tree in memory.
- The model requires a large memory size because the complete XML data stream is mimicked in memory. For memory constrained environments, such as a cell phone, this requirement can be problematic.

The *Xerces2 Java Parser*[3] based on the W3C *Document Object Model* (DOM) [82] is an example of a parser functioning according to the principles of the Tree Model.

[3]More information on Xerces2 Java Parser is available at `http://xerces.apache.org/xerces2-j`.

The Push Model

The second model that emerged after the Tree Model is the Push Model. Its main goal is to address the shortcomings of the Tree Model, namely the high memory requirements and the fact it is unusable for streaming applications. The Push Model states that a compliant parser reads the data stream and for each XML token an event is generated. The event contains implicit and explicit information about the token. Using such a model, the parser pushes the information to the application.

The characteristics of the Push Model are:

- The bootstrap consists of loading and preparing the data stream. Almost no internal data structures must be created during the initialization step.
- Only simple forward navigation is indirectly possible.
- Only token consumption is available.
- The parser is in control, the application does not know when and if an event will be thrown. As such, implementing the application is seen by some as difficult and unnatural.
- Parsing is very fast.
- Very few memory is required because the XML data is not stored in memory. It is up to the application to store the necessary XML data.

The *Xerces2 Java Parser*[3] also supports the *Simple API for XML (SAX)* standard[4], which is an example of a Push Model compliant parser.

The Pull Model

Because the event-oriented programming model is seen as a disadvantage, the Pull Model was developed. Pull Model compliant parsers read only one single token after being instructed by the application. The application can retrieve information about the token from the parser. As such, the information is pulled from the parser by the application.

The characteristics of the Pull Model are:

[4]More information on SAX is available at `http://sax.sourceforge.net`.

- The bootstrap consists of loading and preparing the data stream. Almost no internal data structures must be created during the initialization step.
- Only simple forward navigation is possible.
- Only token consumption is available.
- The application controls and instructs the parser when to act upon the data stream.
- Parsing is very fast.
- Very few memory is required because the XML data is not stored in memory. It is up to the application to store the necessary XML data.

The *XMLPull*[5] parser is an example of this parser model.

The Cursor Model

The Cursor Model is very similar to the Pull Model, but allows random access throughout the XML data stream by directly addressing XML tokens using an XPath expression. The random access makes it possible to target a specific part of the data and, as such, to create a *view* on the data. The XPath expression can for example be used to hide certain irrelevant tokens as such simplifying the token stream. These views can be further examined by the application one token at a time, similar to a Pull Model parser. Hence, the Pull Model can be seen as a forward only and token consumption only version of the Cursor Model.

It is important to note that a specific Cursor Model parser implementation must make a tradeoff between processing speed and memory requirements. Indeed, to obtain fast processing, especially fast XPath navigation, it is necessary to store more information about the XML document into memory.

The characteristics of the Cursor Model are:

- Depending on the specific implementation of the XPath support, bootstrapping is either similar to the Tree Model or to the Push and Pull Models.

[5] More information on the XMLPull parser is available at `http://www.xmlpull.org`.

- All navigation methods are available.
- All token operations (token consumption and token manipulation) are available.
- The application controls the parser.
- Parsing is very fast, but depends on the XPath expression and the parser implementation of the XPath support.
- The memory requirements depend on the specific parser implementation and range from very low to high.

The *Microsoft .NET XPathNavigator*[6] is a parser created according to the principles of the Cursor Model.

The Mapping Model

The last model differs from previous models as it focuses more on the XML content and XML structure than on the XML semantics. First, the model creates object-oriented classes based on the XML data structure. For example, the associated XML Schema or an XML data stream analysis can be used to generate these classes. Second, the XML data, and more specifically the XML content, is mapped to instances of the created object-oriented classes. The application can use the instantiated classes directly just as any other class instance.

The characteristics of the Mapping Model are:

- During the initialization step of the bootstrap, the object classes are instantiated and populated with the XML data.
- All navigation methods are available.
- All token operations (token consumption and token manipulation) are available.
- The application uses the instantiated classes directly.

[6] More information on the Microsoft .NET XPathNavigator is available at `http://msdn.microsoft.com`.

- Bootstrapping is slow because the complete data stream must be completely processed. On top of that, the creation of the classes adds an additional time-cost factor. However, once all data is loaded into the instantiated classes, navigation and token operations are fast.

- This model has medium memory requirements to store the complete XML data in memory, because the classes are optimized to the characteristics of the XML structure and used data types. Hence, the Mapping Model requires less memory than the Tree Model, but more than the Pull and Push models.

The *Microsoft .NET XmlSerializer*[7] is an example of a Mapping Model compliant parser.

It must be mentioned that this model has several issues that prevent a complete and correct mapping of the XML data model, such as the impracticality of handling mixed content and preserving the order of the XML tokens. Due to these issues and the additional time-cost factor for class creation, the mapping model is mostly used for domain-specific applications if the structure of the XML data is known in advance.

4.3 Solving the XML Verboseness

The verboseness issue of XML is more and more being recognized as a potential problem for the adoption of XML in particular fields of applications. Many large and international consortia, like MPEG and W3C, are investigating various solutions.

In this section, we discuss three alternative XML serialization formats in detail, namely ZIP compression, ASN.1, and BiM.

4.3.1 ZIP Compression

The first alternative XML serialization format we discuss in detail is a generic data compression technique, namely ZIP compression. This well known, widespread, and very efficient compression technology is

[7] More information on the Microsoft .NET XmlSerializer is available at `http://msdn.microsoft.com`.

originally developed by PKWARE[8] and is based on DEFLATE [83]. DEFLATE is a combination of the lossless data compression algorithms of Abraham Lempel and Jacob Ziv, and Huffman coding [84].

Abraham Lempel and Jacob Ziv developed the LZ77 [85] and the improved LZ78 [86] algorithms in 1977 and 1978 respectively. Terry Welch extended the LZ78 algorithm resulting in the patented *Lempel-Ziv-Welsh* (LZW) algorithm[9]. Various compression libraries, such as ZLIB [87] and the aforementioned DEFLATE, are based on these techniques. We briefly discuss the basic ideas of these algorithms hereafter. More information can be found in [83–89].

The LZ77 algorithm makes use of a sliding look-back window containing a predetermined number of characters. During the encoding of a data stream, a certain position in the data stream is reached. At that position, the longest subsequent character sequence is searched so this string is also present in the look-back window preceding the current position. If such a sequence is found, the location in the window and the length of the sequence is output and the algorithm continues with the character after the sequence; if not, the character at the current position is output and the algorithm continues with the character at the next position.

The LZ78 algorithm is an improved version of the LZ77 algorithm, mainly reducing the many (time-consuming) string comparisons. It uses a look-up dictionary to achieve compression. Initial, the algorithm starts with an empty dictionary and an empty string. The characters of the data stream are added one by one to the string as long as that string is found in the dictionary. When this is no longer the case, i.e. the string is not present in the dictionary, three things happen: first, the look-up index of the string without the last character in the dictionary is output; second, the last character itself is output; and third, the string is added to the dictionary.

The LZW algorithm is based on the LZ78 algorithm. In contrast to LZ78, the look-up dictionary is not empty at the start of the algorithm, but contains all possible characters of the data stream. The first character of the data stream becomes the initial string and is present in the look-up dictionary. This is no longer the case when the subsequent character is added to the initial string. In general, if a string is not found

[8] More information on PKWARE is available at `http://www.pkware.com`.

[9] More information on LZW can be found in United States Patent 4,558,302, available at `http://www.uspto.gov/patft`.

in the dictionary, again three things happen: first, like in LZ78, the look-up index of the string without the last character in the dictionary is output; second, the new string is added to the dictionary (this is the third step in LZ78); and third, all characters in the string are removed, except the last one. Note that LZW never outputs a character to the output stream, but only indexes.

In contrast to the other serialization techniques, ZIP compression does not use any additional information about XML-based content. In other words, ZIP does not incorporate extra intelligence, for example via an XML Schema, to optimize its compression strategy. Whether or not this is a shortcoming to obtain high compression efficiency will be pointed out in Section 4.5.

4.3.2 Abstract Syntax Notation One

The *International Telecommunication Union* (ITU), and more specifically the *ITU Telecommunication Standardization Sector* (ITU.T) together with the ISO/IEC started development of the ASN.1 standard in 1984 [90]. Originally, it was intended to describe the structure and the type of any kind of organized data for representing, encoding, and decoding of this data in a generic fashion. Currently, it is mainly used in network applications, for example in the *Simple Network Management Protocol* (SNMP), digital certificates, and directory services like the *Lightweight Directory Access Protocol* (LDAP).

Figure 4.1 illustrates the principles of the ASN.1 specification. ASN.1 stores information in an *ASN.1 data message*, which is structured according to an *ASN.1 Schema*. An *original data message* is serialized by an *ASN.1 encoder* in an ASN.1 data message. An *ASN.1 decoder* performs the inverse operation. The *application* that contains the *ASN.1 encoder* and the *ASN.1 decoder* is constructed by compiling standardized *ASN.1 encoding rule(s)* into *original application source code*.

The ASN.1 Schema takes care of the abstract notation of the data by dividing the data into small and simple basic data types – such as integers, booleans, and strings – and by combining these data types into bigger structured types. Listing 4.1 contains an excerpt of the ASN.1 Schemas according to the MPEG-21 DIA-UED standard.

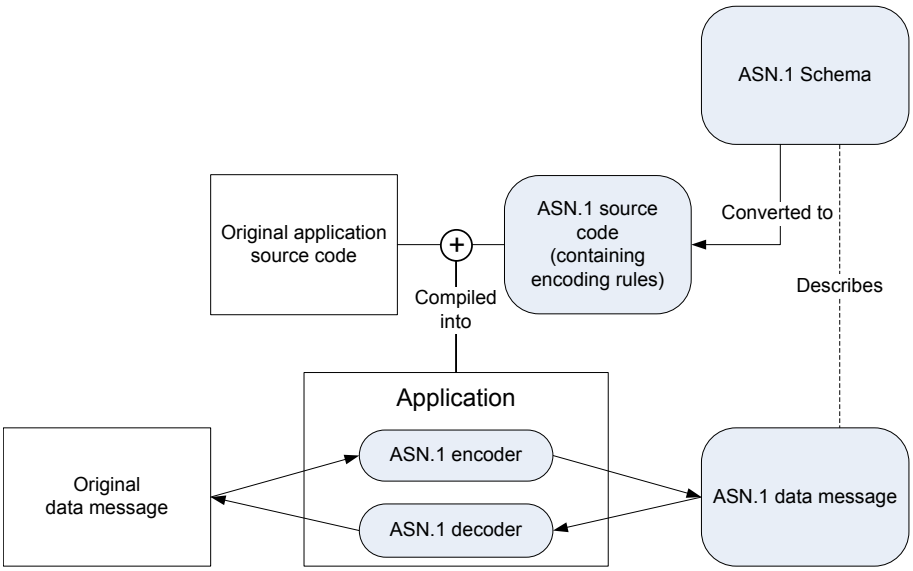


Figure 4.1: Data handling by an ASN.1 application.

Listing 4.1: Excerpt of the ASN.1 Schema of the MPEG-21 DIA-UED standard.

```
/* ######################################### */
/* Definition of Destination                 */
/* ######################################### */
DestinationType ::= SEQUENCE {
 id [NOT NAMESPACE] [ATTRIBUTE] XSD.ID OPTIONAL,
 time [NAME AS CAPITALIZED] Schema-2001.TimeType OPTIONAL,
 location [NAME AS CAPITALIZED] PlaceType OPTIONAL,
 destinationClass [NAME AS CAPITALIZED] SEQUENCE {
  choice [UNTAGGED] CHOICE {
     freeClass-list [UNTAGGED] SEQUENCE OF freeClass
           [NAME AS CAPITALIZED] TextualType,
     stereotypedClass
           [NAME AS CAPITALIZED] ControlledTermUseType
  }
 } OPTIONAL,
 destinationName [NAME AS CAPITALIZED] TextualType OPTIONAL
}
```

The main task of the encoding rule is to remove any redundant structural information from the data and to make sure that the receiver of

the message uses the correct character set or decoder. If a program is aware of the ASN.1 Schema and the selected ASN.1 encoding rule, it can process (read and write) correctly formed data messages. The ASN.1 consortium has standardized the following encoding rules:

- *Basic Encoding Rules* (BER) [91]: the oldest ASN.1 encoding method serializes the data message using a *Key-Length-Value* (KLV) system. The *Key* (also called the *Tag*) is used to identify the data type, *Length* expresses the number of bytes the value requires, and *Value* contains the actual data of the structure.
- *Distinguished Encoding Rules* (DER) [91]: DER is a variation of BER. It defines a more strict system that eliminates all options and choices such that the original data message can only be encoded into one unique ASN.1 data message. As such, DER is very suitable to digitally sign information as only one form to represent the data is possible.
- *Canonical Encoding Rules* (CER) [91]: equal to DER, but with the ability to start processing a KLV triplet before it is completely loaded into memory.
- *Packed Encoding Rules* (PER) [92]: PER is also based on BER, but only stores the *Key* if this information cannot be derived from the ASN.1 Schema. Also, the *Length* is not stored as long as its value can be deduced from the used data type. Currently, PER is the most optimized encoding rule in terms of compactness.
- *XML Encoding Rules* (XER) [93]: the name may give the impression it is only intended for the coding of XML data, however this is not the case. In fact, the encoding rule serializes any data message as plain-text XML. It uses the data type name as the XML element name and its value as the XML element value.

Table 4.2 shows the results of a boolean value "true" that is encoded by the different encoding rules. It is clear that PER reduces most overhead.

Recently, the ASN.1 standard was extended to improve support for XML data, as illustrated in Figure 4.2, by defining mapping rules between an XML Schema and an ASN.1 Schema [94]. As such, the ASN.1 Schema can be seen as a full alternative to XML Schema.

Based on the ASN.1 Schema, different tools are available to automatically generate source code that can be compiled into other applications.

Table 4.2: ASN.1 encoding rules: results of encoding the boolean value "true."

Encoding Rules	Result for a boolean value "true"
BER	$(010101)_{16}$ or $(010102)_{16}$ or ...
DER & CER	$(0101FF)_{16}$
PER	$(1)_2$
XER	`<xer:BOOLEAN>true</xer:BOOLEAN>`

These tools also generate source code for the different encoding rules so for each encoding rule two methods are available: an encoding method and a decoding method. The former accepts a valid XML file and returns the ASN.1 data message; the latter accepts an ASN.1 data message and returns a valid XML file. As such, it is trivial to process and generate ASN.1 data messages compliant for a given XML Schema from any application. ASN.1 in fact creates a specialized Mapping Model parser for a particular XML Schema.

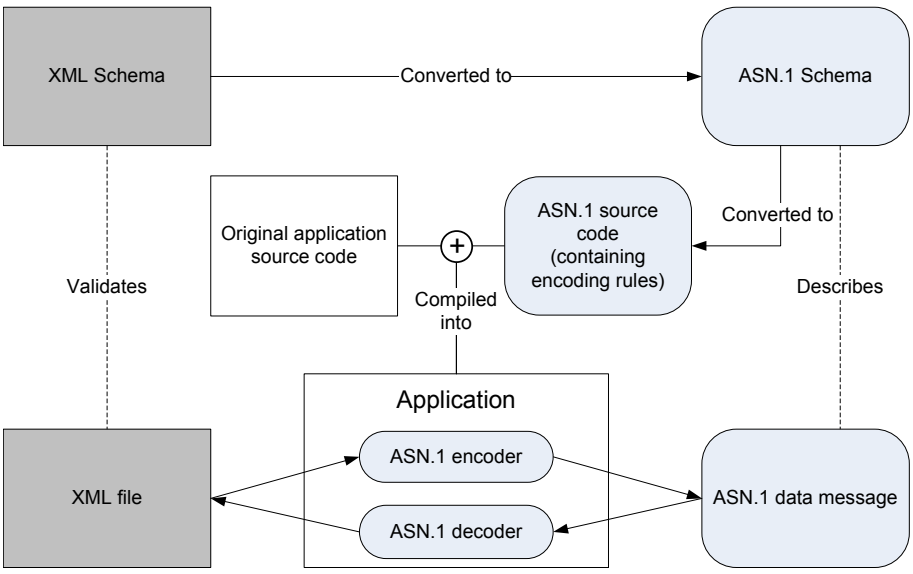


Figure 4.2: Handling XML documents by an ASN.1 application.

4.3.3 Binary MPEG Format for XML

The MPEG group developed an alternative XML serialization format in 2001 as part of the MPEG-7 Part 1 – *Systems* standard (the *Binary format for Metadata*) [49,95]. Initially, it was intended as an alternative and compact serialization of the XML-based MPEG-7 descriptions. According to [95], this approach can also be applied to XML-based data in general as long as there is an *MPEG-7 Description Definition Language* (DDL) Schema [96] or a W3C XML Schema available. The DDL adopted the XML Schema specification with some multimedia-related extensions. Hence, XML Schema is a subset of DDL. Recently, the MPEG-7 Binary format for Metadata technology has been relocated to a new MPEG standard formally known as MPEG-B Part 1 – *Binary MPEG Format for XML*, keeping the original abbreviation BiM [97].

BiM is an XML Schema aware encoding scheme for XML documents, i.e., it uses information from the XML Schema to create an efficient alternative serialization of XML documents within the binary domain. This knowledge enables the removal of structural redundancy (e.g., white space, and element and attribute names) thus achieving high compression ratios with respect to the document structure. Furthermore, element and attribute names as well as data are encoded using dedicated encoders based on the data type (e.g., integer, float, and string), which further increases the compression ratio. The advantages of BiM over traditional plain-text compression algorithms are the support of parsing the XML data in the binary domain (thus without decoding to plain-text XML), its streaming capabilities, and dynamic and partial updating of existing XML trees.

To achieve the latter, BiM can divide an XML tree into different parts. Each part is encoded into an *access unit* and contains optional *schema update units* and one or more *fragment update units*. As a decoder needs to know the set of utilized XML Schemas in order to decode the XML data, the schema update unit makes it possible to modify the initial set of schemas. More interesting are the fragment update units, which in turn consist of three parts: a *fragment update command*, a *fragment update context*, and a *fragment update payload*. The fragment update command specifies the decoder action for the corresponding fragment and can be either add, delete, replace, or reset, i.e., BiM provides partial updates of an XML document. The fragment update context is used to uniquely determine the location of the fragment to be updated in the

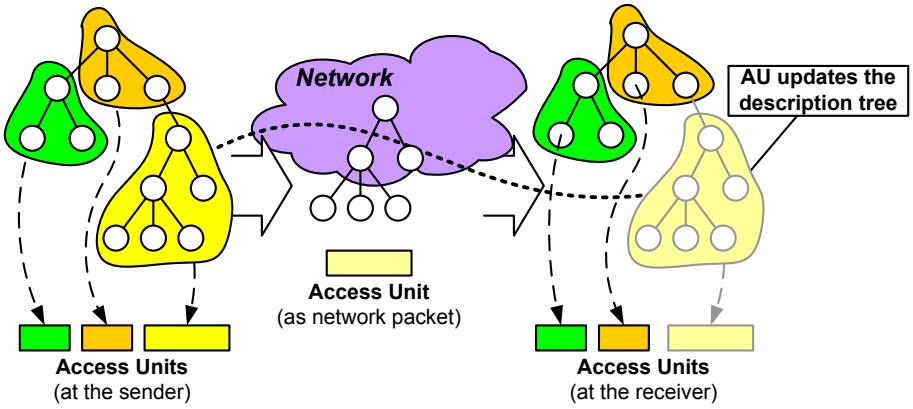


Figure 4.3: Streaming XML Documents over the Network using *Access Units* (AUs).

XML document by, for example, an XPath expression. Finally, the fragment update payload contains the actual encoded XML data of the update. Figure 4.3 illustrates how an XML document is divided into access units and streamed over the network. In particular, it shows how a subtree of the whole XML document is transmitted over the network and added to the XML tree at the receiver side (i.e., the dotted line). BiM is the only standardized serialization format that natively supports XML updates.

4.4 Serialization-Agnostic Parser

This section defines an XML parser that is capable of handling textual XML as well as non-textual serialized XML data (Figure 4.4). On top of that, our parser is able to transparently handle updates of XML data, for example time-varying XML data as described in the previous chapter. Hence, our XML parser supports and implements an update functionality making it possible to modify parsed XML data on the fly. First, we determine the optimal XML Parser Model.

Because we want to support update functionality, all parser models offering only simple forward navigation cannot be used. This means that the Push and Pull Models are not suitable. Indeed, keep the possibility in mind that updates of the XML data can occur anytime. In other

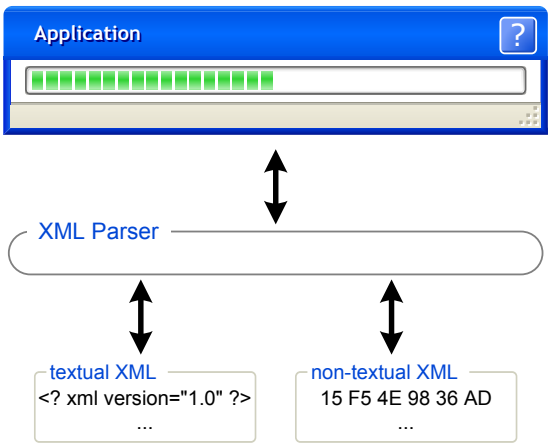


Figure 4.4: Overview of a serialization-agnostic XML parser: the application is unaware of the kind of XML serialization.

words, updates can modify information that has already been processed by the parser. Parsers with only forward navigation cannot handle these changes because the application has no access to the information that is updated. Even if the application is informed that the data are changed, it is necessary to restart the parsing.

The Mapping Model seems to provide the best solution because the updates can manipulate the instantiated classes directly in a fast and straightforward way. Thus, the application is immediately aware of the modified information. Unfortunately, a parser compliant to the Mapping Model is built based on one or more specific XML Schemas, hence it is only usable for XML data valid to the given XML Schema. As such, this solution is advised if the XML Schemas are known during the development of the parser. This is especially applicable for standardized and normative schemas. Nevertheless, we want to develop a generic XML parser; hence, we will not use this model.

From the two remaining XML parser models, the Cursor Model has a slight advantage over the Tree Model as it has XPath handling capabilities. If an update is accompanied by an XPath expression, the update location can be found immediately by the parser by reusing this internal capability. Moreover, the same XPath expression can also be used to signal the application that an update took place at the given location.

Hence, the Cursor Model is our preferred parser model for the creation of

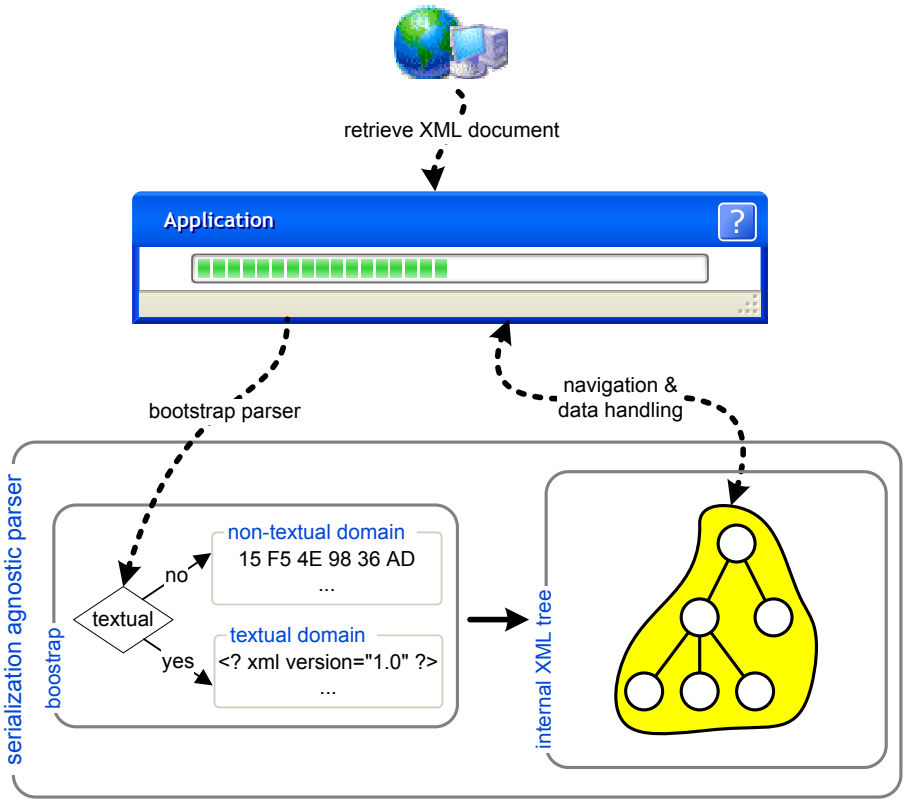


Figure 4.5: Architectural overview of our serialization-agnostic XML parser based on the Cursor Model.

the desired XML parser. We have chosen to optimize our parser for fast XPath evaluation over memory compactness. The architectural overview when using a Cursor Model parser is given in Figure 4.5. In practice, an application retrieves an XML document from a source and sends it to the bootstrap method of our parser. The bootstrap method determines the serialization format of the XML data by looking at the extension of the filename, or more advanced using the MIME-type information. Once the bootstrap method has identified the serialization type, it can handle the data appropriately. If the content encoding format of the XML data is textual, it is only necessary to create the internal XML tree for which existing XML Cursor Model parsers can be used. If the XML data are encoded by a non-textual serialization format, the appropriate decoder is used. If BiM is used, it is possible to create the internal

tree representation directly during decoding; for the ZIP compression and ASN.1-PER encoding, the data stream must be completely decoded before the creation of the internal XML tree can start.

In case of an update to a previously received XML data stream, no special precautions need to be provided for BiM because it natively supports updates. For the three other serialization types, it is necessary to explicitly indicate the fact it is an update. This is done during the bootstrap method by adding an XPath expression that specifies the location of the update. As such, it is possible to enable the update functionality even for the serialization formats that do not natively support this. In other words, as the BiM technique natively supports XML updates, we created application specific update functionality also for the remaining serialization techniques, in particular plain text, ZIP compression, and ASN.1-PER encoding.

4.5 Evaluation

The goal of this section is to evaluate the usefulness of the alternative XML serialization formats in comparison to the regular plain-text serialization method. Besides the investigation of the compression efficiency, we also measure the time required to serialize and to parse the XML data using the different serialization methods. Two use cases are developed for the evaluation as explained hereafter.

4.5.1 Use Case 1: Usage Context Negotiation

The first use case is based on a part of the context negotiation scenario as described in the previous chapter. A client device uses the MPEG-21 DIA-UED software toolkit discussed in Section 2.3.2 to structure its initial usage context. This information is sent to the broker. Next, time-varying metadata are transmitted with updates to the initial usage context.

4.5.2 Use Case 2: Really Simple Syndication

RSS is an XML-based application that enables users to be informed when an update occurs to an Internet news source. The UML class model of the RSS specification is depicted in Figure 4.6. RSS defines a

container structure: an RSS document – the *RSS feed* – contains one or more *Channels*, e.g., a weather information channel. A Channel stores, among other things, information about the topic and combines multiple *Items*. One Item holds a particular news piece on the topic of the given Channel, e.g., the weather information for a particular city. An *RSS viewer* retrieves the RSS feed from a server on a regular basis, e.g., by downloading the RSS document every day. If new Items are available, the RSS viewer informs its user about these by, for example, showing the titles of the new Items on his or her display. The RSS publisher updates the RSS feed by adding new Items or removing obsolete Items at the server. Normally, the Channel information is not modified.

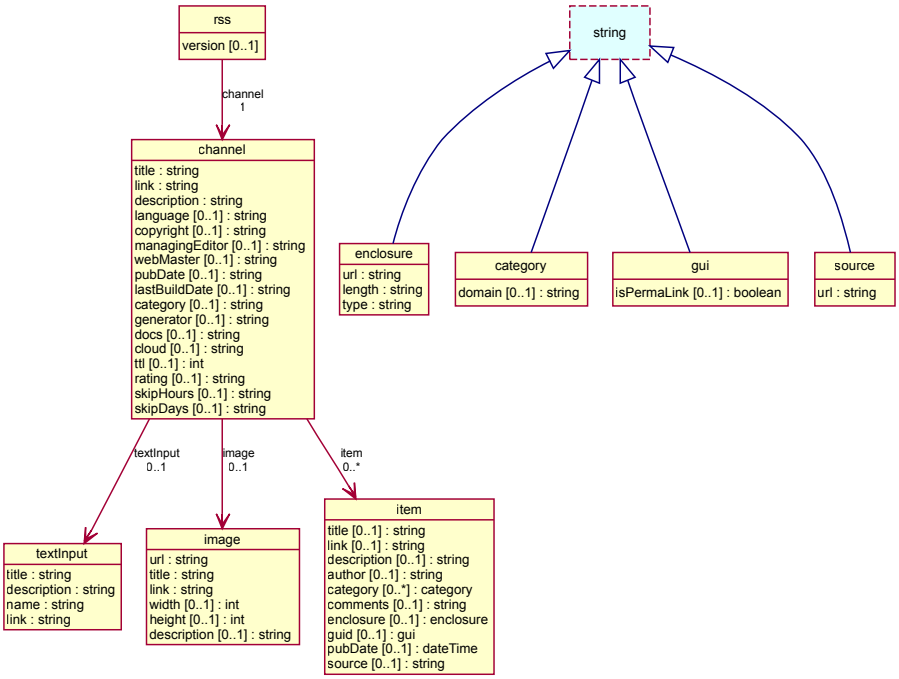


Figure 4.6: The RSS feed structure.

Recently, RSS became a popular tool as a way to publish *podcasts*. Indeed, a podcast publishes multimedia data (such as audio and video) using RSS Feeds, such that the feed subscribers automatically receive new content. This is accomplished by adding the optional `enclosure` element to an RSS Item, which contains an URL to the actual location of the audio-visual data stream. As such, it is comparable to the technique

of adding multimedia content to a Web page.

The issues of the RSS application as it is used today are mainly related to the way the RSS viewer checks the Internet news source for any modification. Indeed, the RSS viewer regularly downloads the RSS data file anew to verify if Items were added. If the RSS data file was not changed since the last retrieval, this download is unnecessary and the bandwidth waste is obvious. But even if new Items are present in the RSS file, bandwidth is still wasted, because the new Items are usually only a relatively small part of the complete XML file. Similar to the problems explained in Chapter 3, this overhead is expensive for the end user, especially when using an Internet connection that is paid for on a per byte basis. For content providers, the overhead can also become a big concern as the overhead becomes significant if millions of users are subscribed to a particular feed. Note that content providers also pay a fee for the bandwidth used. In other words, the overhead introduces a cost for the end user as well as for the content provider.

This overhead can be addressed by creating a (proprietary) update functionality for RSS. Our solution appends the date and time of the last synchronization to the HTTP download request of the RSS viewer. As such, the RSS provider only needs to send the new or modified Items.

4.5.3 Methodology

For the evaluation of the first use case, we created a realistic initial usage context and three updates, all of them compliant to MPEG-21. The three updates differ in size and target different parts of the usage context. The first update modifies the information about the bandwidth of the network. The second update changes the terminal's display information. And finally, the third update is a large update that changes the user and the natural environment information. The listings of the initial context and the updates are shown in Listings A.1 to A.4 of Appendix A.

For the second use case, we emulate the typical usage of the RSS application. An RSS viewer retrieves the RSS data from a content provider daily during the period of one month. In practice, we used the MSDN[10] RSS feed of the month November 2004 containing a total of 53 Items

[10]The *Microsoft Developer Network* (MSDN) Website is available at `http://msdn.microsoft.com`.

scattered over the weekdays. A part of the RSS feed can be found in Listing 4.2.

Listing 4.2: Excerpt of the Microsoft Developer Network RSS feed of the month November 2004.

```
<?xml version="1.0" encoding="utf-8"?> <rss version="2.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:mmlab:be:rss20">
 <channel>
  <title>MSDN Just Published</title>
  <link>http://msdn.microsoft.com/</link>
  <description>Keep current with all the new technical
      articles, columns, specifications, and resources
      published on the Microsoft Developer Network (MSDN).</
      description>
  <language>en-us</language>
  <pubDate>2004-11-24T22:11:49</pubDate>
  <lastBuildDate>2004-11-24T22:11:49</lastBuildDate>
  <ttl>1440</ttl>
  <item>
    <title>.NET in the Real World: Code Generators with .NET
        </title>
    <description>You're already using code generators
        whether you know it or not; [...]</description>
    <link>http://msdn.microsoft.com/vstudio/using/columns/
        realworld/default.aspx</link>
    <category domain="msdndomain:DevLangVers">ASP</category>
    <pubDate>2004-11-04T08:00:00</pubDate>
  </item>
  <item>
    <title>.NET Rocks! - Juval Lowy on .NET 2.0</title>
    <description>Juval Lowy joins Carl, Rory, and the gang
        this week for a romp [...]</description>
    <link>http://msdn.microsoft.com/dotnetrocks/</link>
    <category domain="msdndomain:ContentType">Multimedia</
        category>
    <pubDate>2004-11-03T08:00:00</pubDate>
  </item>
  [...]
 </channel>
</rss>
```

The XML data of the two use cases are marshalled in four ways:

- *Plain text:* this is the classical way of marshalling XML data, namely as plain text. For these tests, we use the UTF-8 notation,

which is the most common format for XML data. The XML data are sent to the receiving application, i.e., the broker in Use Case 1 and the RSS viewer in Use Case 2. The receiving application passes the data to the bootstrap method of our created parser. The parser creates the internal XML tree for the given XML data.

- *ZIP:* the second method to serialize the XML data is by compressing it before transmission. For the evaluation, we use the internal ZIP compression functionality of *Java 2 Standard Edition* (J2SE) version 5.0, which is based on DEFLATE (see Section 4.3.1). The bootstrap method of our parser accepts these ZIP-compressed XML data and decompresses it into plain text before starting the creation of the internal XML tree.

- *ASN.1-PER:* this serialization type applies the Abstract Syntax Notation One Packed Encoding Rules, as discussed in Section 4.3.2, to the textual XML data. To apply this serialization type, the DIA-UED and RSS PER encoders and decoders are created as follows. First, the ASN.1 Schemas are generated using the MPEG-21 DIA-UED and RSS XML Schemas[11]. These generated schemas are used to generate Java source code[12]. Next, the code is compiled together with a newly developed application and results in a new software module (see also Figure 4.2). This software module accepts an XML file (valid to the particular MPEG-21 or RSS schemas) as input and returns the ASN.1-PER encoded version and vice versa. There are two independent ASN.1-PER software modules: one module for Use Case 1 (based on the DIA-UED XML Schemas) and one module for Use Case 2 (based on the RSS XML Schema). These software modules are linked to our parser, nevertheless, it is necessary to fully decode the ASN.1 data messages to plain text before the creation of the internal XML tree can start.

- *BiM:* this last alternative XML serialization format uses the Binary MPEG Format for XML, as described in Section 4.3.3. BiM is applied to the textual XML data, resulting in BiM-encoded XML. The parsing of this BiM-encoded data occurs directly by our developed parser, thus the internal XML tree is constructed simultaneously with the processing of the BiM serialized data. This is

[11] XML Schema to ASN.1 Schema conversion was done by the online translation tool of the ASN.1 information Website, available at `http://asn1.elibel.tm.fr/xsdasn1`.

[12] Generation of Java source code from the ASN.1 Schemas was done by the OSS Nokalva ASN.1 tools, available at `http://www.oss.com`.

accomplished by integrating the BiM reference software code [98] with our own serialization-agnostic parser. As a result, we do not need to decode the data first before the creation of the internal XML tree can start.

All four serialization methods are used in two modes, namely a *Full mode* and an *Update mode*:

- *Full mode:* this is the classical way of exchanging XML-based information, in particular by transmitting the complete and well-formed XML file valid to the corresponding XML Schemas. For Use Case 1, the time-varying XML updates are applied to the usage environment description on the client device and the resulting description is used as input for the serialization type. For the Use Case 2, an RSS feed is constructed containing the Items published up to the "current day" of the month.
- *Update mode:* in this mode, only the modifications are transmitted. In other words, the three updates for the first use case and the Items that were published between two successive days for the second use case. Note, only the BiM serialization type can natively handle updates to the previously received XML information through its Access Units (as discussed in Section 4.3.3). For all other serialization types, a proprietary, and therefore non-interoperable, construction is created to handle the update information. Our serialization-agnostic XML parser has hard-coded rules to process these updates. As such, this mode is only for the BiM serialization generic, interoperable, and applicable for commercial and enterprise applications. Nevertheless, the update mode for the plain text, ZIP, and ASN.1-PER serialization types is useful for comparison.

For each use case, each serialization type, and each mode, we measured the byte size of the data that must be transmitted, the time required to parse the data, and – if applicable – the time required to prepare the XML data before transmission. The latter implies the time to compress the data for the ZIP serialization, to execute the ASN.1-PER encoding software for the ASN.1-PER type, and to use the BiM reference software encoder for the BiM serialization. The time required to parse the data is the time needed for our serialization-agnostic parser to create the

internal tree and the time required to decompress the ZIP-compressed XML data and the time required to decode the PER-encoded data using the generated ASN.1-PER decoding software.

We executed hundred runs of the tests without exiting the Java Runtime Environment. The results in Section 4.5.4 show the average thereof with one outlier per test discarded. This outlier is caused by a run of the Java garbage collector.

All measurements were performed on a low-end device, namely a laptop computer equipped with an Intel Centrino Pentium M 1.1Ghz processor running Windows XP Pro SP2 and J2SE version 5.0. For the runtime measurements we use the *System.nanoTime()* method of J2SE version 5.0.

4.5.4 Results and Discussion

Use Case 1: Usage Context Negotiation

The results of the first use case are shown in Table 4.3 and Table 4.4. The tables show for each of the four serialization types the byte size of the data, the time required to parse this file, and the time required to prepare the XML data. The latter value is found in the column denoted with "prep."

In case of the full mode (Table 4.3), the results clearly show that ZIP compression doubles the parsing time compared to the plain-text serialization and achieves a compression ratio of 4:1. ASN.1-PER triples execution time and obtains a compression ratio of 5:1. And finally, the time required to parse BiM-encoded XML data is about 90 times higher than plain-text XML data while at the same time the byte size is about ten times smaller.

In case of the update mode (Table 4.4), the BiM compression efficiency decreases from 10:1 to 6:1, but the parsing time increases; BiM now needs a huge time longer to parse the data in comparison to plain-text serialization. Also, ZIP is less efficient, namely 3:1 and requires four times more time to parse. Contrary to the other binary serialization types, ASN.1-PER improves its compression efficiency, namely from 5:1 for the full mode to 10:1 for the update mode, without a time penalty. This improvement is a result of the very high compression efficiency for the first and second update (see Listings A.2 and A.3 in Appendix A).

Because these updates do not contain any free-form text, with the exception of the `id`-attribute value, ASN.1-PER is able to create a very condensed alternative serialization of the XML-based data that even outperforms the two other methods.

The time to serialize the XML data for the different serialization methods reveals similar results: BiM is by far the slowest serialization type, then ASN.1-PER, and finally, ZIP compression.

The slow parsing of the BiM-encoded XML data and the slow BiM serializing of the textual XML data can be explained by the usage of the MPEG reference software for BiM encoding and decoding, which is not optimized in terms of execution time. Commercial implementations of the MPEG-B standard should provide a more optimized solution in the future, unfortunately such a solution is not yet available. Also, the XML Schema for MPEG-21 DIA-UED is very complex and comprehensive. Analysis of this schema is not straightforward and introduces a time penalty for parsing and encoding. Optimized implementations of the BiM standard will probably provide a caching mechanism for analyzed XML Schemas. Furthermore, as the MPEG-21 DIA-UED XML Schemas are standardized, it is envisaged that applications or devices will use a hard-coded version thereof avoiding the analysis phase of the XML Schema. The ASN.1-PER serialization avoids this pitfall as the analysis phase of the XML Schema is performed during creation of the Java source code. ASN.1-PER requires over 200 Java classes to model the UED XML Schemas, in other words over 200 complex type XML elements are used in the schemas. The ASN.1-PER tools require about 400 milliseconds to create these classes.

In Table 4.5, Figure 4.7, and Figure 4.8, the cumulated byte sizes for the different serialization types are listed. These are the total number of bytes that are sent when the three updates are consecutively applied to the initial usage environment description. As such, this is the amount of data that must be transmitted over the network to inform the broker service about the context.

The reduction when using an alternative serialization format for XML data is the highest for the BiM method with all its features exploited, in other words, for BiM serialization in update mode.

	Plain Text		ZIP			ASN.1-PER			BiM		
	size (bytes)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)
Full UED	8,805	3	2,175	4	6	1,808	83	9	939	628	276
First update	8,810	3	2,179	4	6	1,806	49	9	937	627	275
Second update	8,574	3	2,162	4	9	1,785	57	9	933	627	275
Third update	13,847	4	2,790	5	7	2,450	61	13	1,046	654	279

Table 4.3: Results for Use Case 1: Usage Context Negotiation – Full mode.

	Plain Text		ZIP			ASN.1-PER			BiM		
	size (bytes)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)
Full UED	8,805	3	2,175	5	7	1,808	83	10	939	631	277
First update	710	1	524	3	3	45	9	2	305	681	265
Second update	808	1	526	2	6	55	7	2	697	680	267
Third update	10,393	3	2,107	4	7	1,655	49	11	1,110	1,050	387

Table 4.4: Results for Use Case 1: Usage Context Negotiation – Update mode.

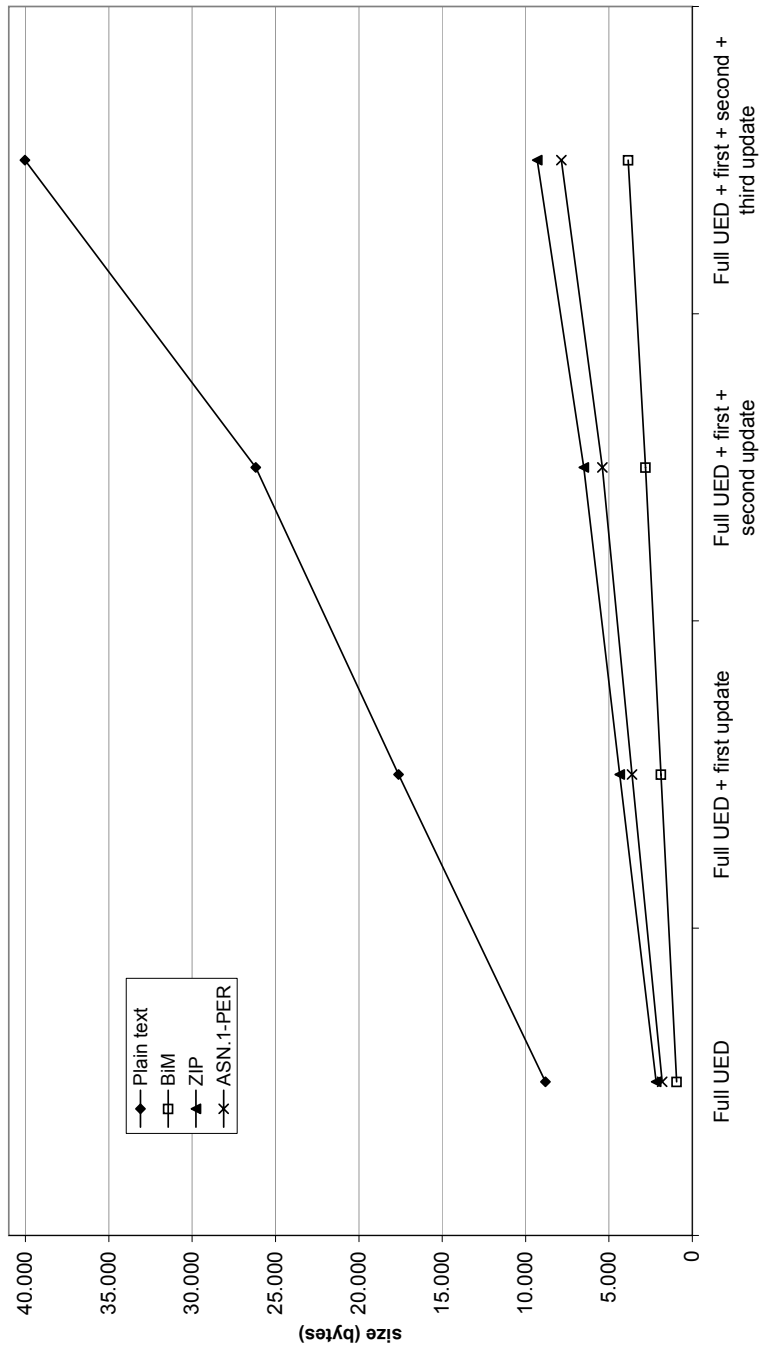


Figure 4.7: Cumulated byte size Use Case 1: Usage Context Negotiation – Full mode.

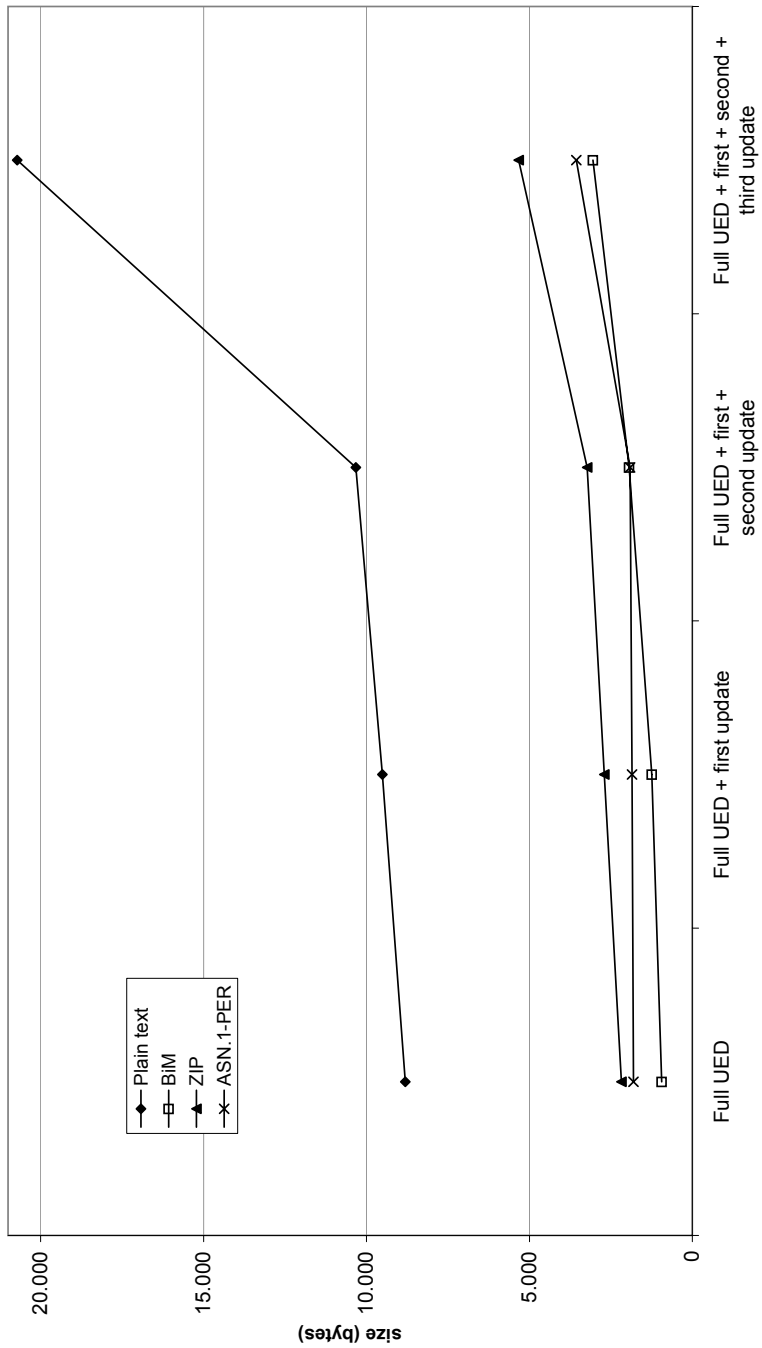


Figure 4.8: Cumulated byte size Use Case 1: Usage Context Negotiation – Update mode.

Table 4.5: Cumulated byte size for Use Case 1.

	Full		Update	
	size (bytes)	reduction (%)	size (bytes)	reduction (%)
Plain text	40,036		20,716	48.25
ZIP	9,306	76.75	5,332	86.68
ASN.1-PER	7,849	80.39	3,563	91.10
BiM	3,855	90.37	3,051	92.37

Use Case 2: Really Simple Syndication

The results of the second use case are listed in Table 4.6 and Table 4.7. These tables show for each of the four serialization types the byte size of the data, the time required to parse this file, and the time required to prepare the XML data. The latter value is found in the column denoted with "prep."

In case of the full mode (Table 4.6), the results show that the time required to parse BiM-encoded XML data is about 20 times slower than plain-text XML data, while at the same time the byte size is about five times smaller. The compression efficiency is lower than in the first use case because the XML data contains more string values whereas in Use Case 1 there is more overhead due to the XML structure. This fact also results in the lower compression efficiency for the ASN.1-PER serialization, namely 3:2. Only ZIP retains its compression ratio of 4:1. The parsing and the encoding of the XML data for the different serialization types are proportional to the byte size. The parsing of the BiM-encoded XML data is faster in comparison to the first use case due to the simple XML Schema for RSS feeds. Nevertheless, the parsing time remains high compared to the other serialization types. The ZIP serialization is, again, the fastest of the three alternative serialization types.

Table 4.7 shows the results in the Update mode. The "–" value indicates that there is no new Item for the given day, so no data is transmitted. We notice, together with larger byte size values in Table 4.6, that the serializing and parsing timings for the BiM serialization method is higher than in the first use case. This can be explained as for every new Item in the RSS feed, a new fragment update unit is created. The parsing and serialization of multiple fragment update units into one access unit

decreases the processing speed. Once again, we want to emphasize that we use the non-optimized reference software and that a more optimized implementation of the BiM specification, which handles multiple fragment update units better, will reduce the executing time for parsing as well as serializing. The results also show that the gap of the compression efficiency for the different serialization types is closing. In fact, ZIP compression outperforms BiM encoding with respect to parsing and encoding – mainly due to performance issues of the BiM reference software – but regarding byte size BiM is superior in all cases with a few exceptions though. In particular, if many new RSS Items need to be transmitted, ZIP provides a slightly better compression ratio than BiM. This indicates that the overhead for creating a new fragment update unit for every new Item is large, not only in terms of execution time, but also in terms of bytes. Nevertheless, we would like to remind the reader that BiM natively provides the update functionality, while for ZIP compression we implemented application domain-specific rules in our parser to achieve the same effect.

The cumulated byte sizes for the different serialization types are listed in Table 4.8 and depicted in Figure 4.9 and Figure 4.10. We compare the used bandwidth after one month with regard to the different serialization types.

Once again, the reduction is the highest for BiM serialization in update mode. However, ZIP is only slightly worse than BiM for both the full and update modes, but the latter mode implies application domain-specific handling of the data.

4.6 Related Work

Before concluding this chapter, we want to give an overview of other activities currently undertaken to solve the verboseness issues of XML data.

The W3C, the driving force behind XML, recognizes the verboseness concerns and founded a task force in 2004 to investigate the usefulness and desirability of an alternative serialization format. This resulted in a first working draft of relevant use cases and applications that could benefit from a non-textual serialization. This document is regularly updated and can be found on the Website of the W3C [99]. The report is used to determine if it is possible to select one specific kind of alter-

Table 4.6: Results for Use Case 2: Daily retrieval of an RSS feed – Full mode.

	Plain text		ZIP			ASN.1-PER			BiM		
day	size (bytes)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)
1	611	1	503	27	5	280	21	2	226	237	65
2	1,217	1	762	27	4	687	23	2	437	246	66
3	1,748	1	845	28	4	1,018	26	3	522	248	66
4	3,207	2	1,224	30	5	2,015	31	4	895	262	67
5	4,013	2	1,372	31	7	2,564	34	5	1,049	273	68
6	4,013	2	1,372	31	5	2,564	33	5	1,049	272	68
7	4,013	2	1,372	31	5	2,564	35	5	1,049	272	67
8	5,372	2	1,724	33	6	3,495	41	6	1,403	279	68
9	5,372	2	1,724	35	5	3,495	41	6	1,403	279	68
10	8,783	3	2,403	37	7	5,905	55	9	2,120	296	70
11	9,703	4	2,691	38	7	6,599	59	10	2,402	301	71
12	11,417	4	2,973	41	8	7,741	67	11	2,698	372	72
13	11,417	4	2,973	42	8	7,741	65	11	2,698	372	72
14	11,417	4	2,973	41	7	7,741	65	11	2,698	372	72
15	12,398	5	3,121	42	8	8,377	70	12	2,856	439	73

Table 4.6: Results for Use Case 2: Daily retrieval of an RSS feed – Full mode (continued).

	Plain text		ZIP			ASN.1-PER			BiM		
day	size (bytes)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)
16	15,311	6	3,514	45	9	10,281	80	14	3,306	895	74
17	17,905	7	3,938	50	11	12,157	90	16	3,732	1,043	76
18	25,526	9	4,916	59	13	17,092	120	23	4,822	1,041	90
19	27,285	10	5,137	62	15	18,249	128	25	5,064	1,278	97
20	27,285	10	5,137	61	13	18,249	126	25	5,064	1,278	98
21	27,285	10	5,137	61	14	18,249	126	25	5,064	1,277	97
22	29,497	10	5,615	65	14	19,776	134	26	5,533	1,292	99
23	30,212	11	5,744	65	16	20,263	137	26	5,656	1,296	99
24	30,212	11	5,744	65	15	20,263	138	27	5,656	1,298	99
25	30,952	11	5,977	66	16	20,831	139	27	5,886	1,305	99
26	31,606	11	6,078	67	15	21,286	139	28	6,006	1,304	100
27	31,606	11	6,078	67	17	21,286	138	28	6,006	1,305	100
28	31,606	11	6,078	67	15	21,286	137	28	6,006	1,307	100
29	33,946	12	6,457	70	17	22,971	145	30	6,396	1,316	101
30	39,669	14	7,303	77	18	27,075	165	34	7,291	1,747	103

Table 4.7: Results for Use Case 2: Daily retrieval of an RSS feed – Update mode.

	Plain text		ZIP			ASN.1-PER			BiM		
day	size (bytes)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)
1	611	1	548	27	5	280	21	2	226	237	65
2	1,433	1	819	27	3	707	31	3	364	1863	66
3	1,358	1	787	26	4	631	36	3	330	1864	66
4	2,286	1	1,061	28	5	1,297	39	4	818	3288	120
5	1,633	1	875	27	4	849	30	4	425	1860	66
6	–	–	–	–	–	–	–	–	–	–	–
7	–	–	–	–	–	–	–	–	–	–	–
8	2,186	1	1,044	27	3	1,231	33	4	785	3281	120
9	–	–	–	–	–	–	–	–	–	–	–
10	4,238	2	1,485	31	5	2,710	46	6	1,969	8087	283
11	1,747	1	984	28	4	994	38	4	519	1883	67
12	2,541	1	1,064	29	4	1,442	35	5	862	3329	122
13	–	–	–	–	–	–	–	–	–	–	–
14	–	–	–	–	–	–	–	–	–	–	–
15	1,808	1	884	27	4	936	32	4	427	1865	67

Table 4.7: Results for Use Case 2: Daily retrieval of an RSS feed – Update mode (continued).

	Plain text		ZIP			ASN.1-PER			BiM		
day	size (bytes)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)	size (bytes)	prep. (ms)	parse (ms)
16	3,740	2	1,211	30	4	2,204	41	6	1,338	4927	178
17	3,421	2	1,314	29	5	2,176	40	5	1,389	5013	178
18	8,448	3	1,809	36	6	5,235	64	11	4,208	19221	682
19	2,586	1	1,064	28	5	1,457	36	5	861	3352	126
20	–	–	–	–	–	–	–	–	–	–	–
21	–	–	–	–	–	–	–	–	–	–	–
22	3,039	1	1,299	28	4	1,827	45	5	1,266	5000	183
23	1,542	1	860	26	4	787	30	4	405	1714	70
24	–	–	–	–	–	–	–	–	–	–	–
25	1,567	1	925	27	4	868	31	4	469	1746	70
26	1,481	1	848	27	4	755	30	4	396	1684	70
27	–	–	–	–	–	–	–	–	–	–	–
28	–	–	–	–	–	–	–	–	–	–	–
29	3,167	1	1,202	28	4	1,985	40	5	1,287	5082	185
30	6,550	3	1,775	33	5	4,404	58	8	3,168	13008	475

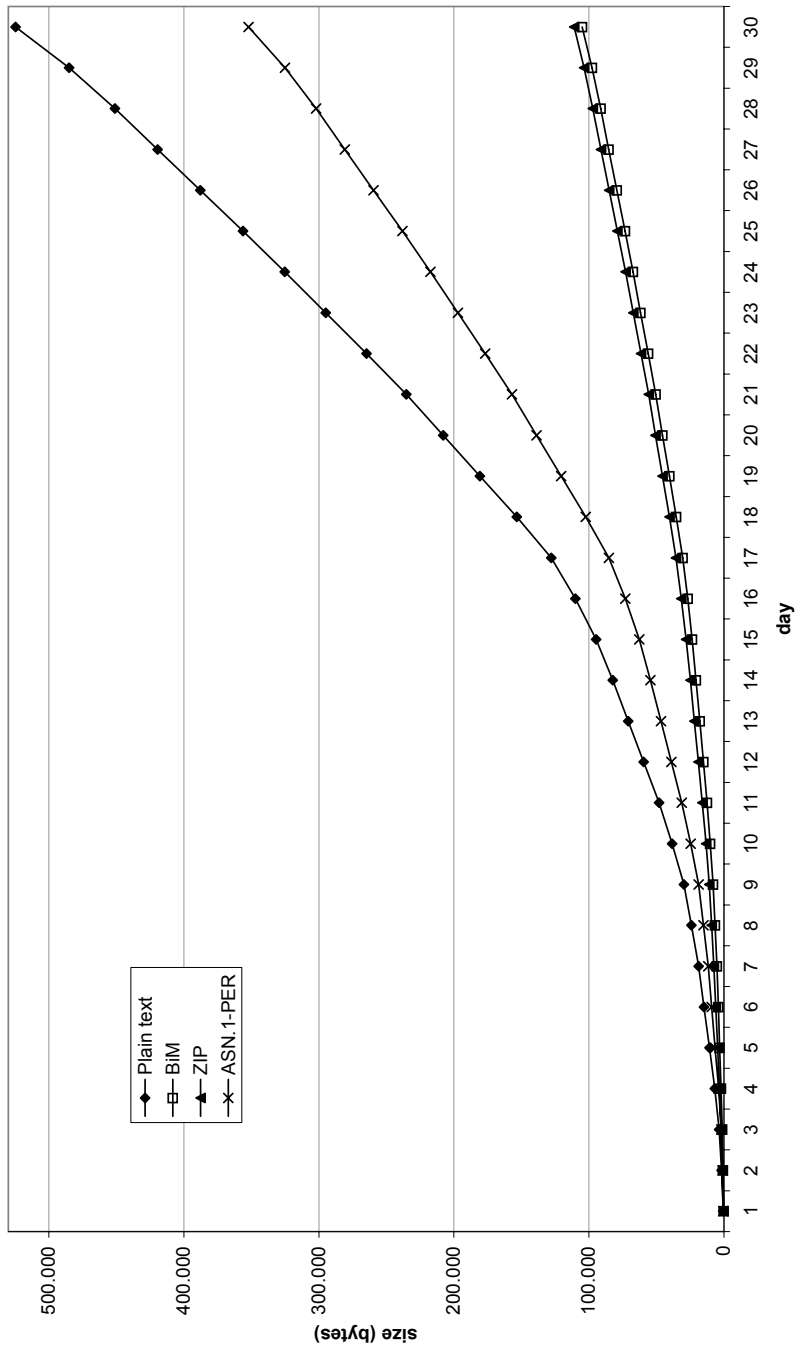


Figure 4.9: Cumulated byte size for Use Case 2: daily retrieval of the RSS feed – Full mode.

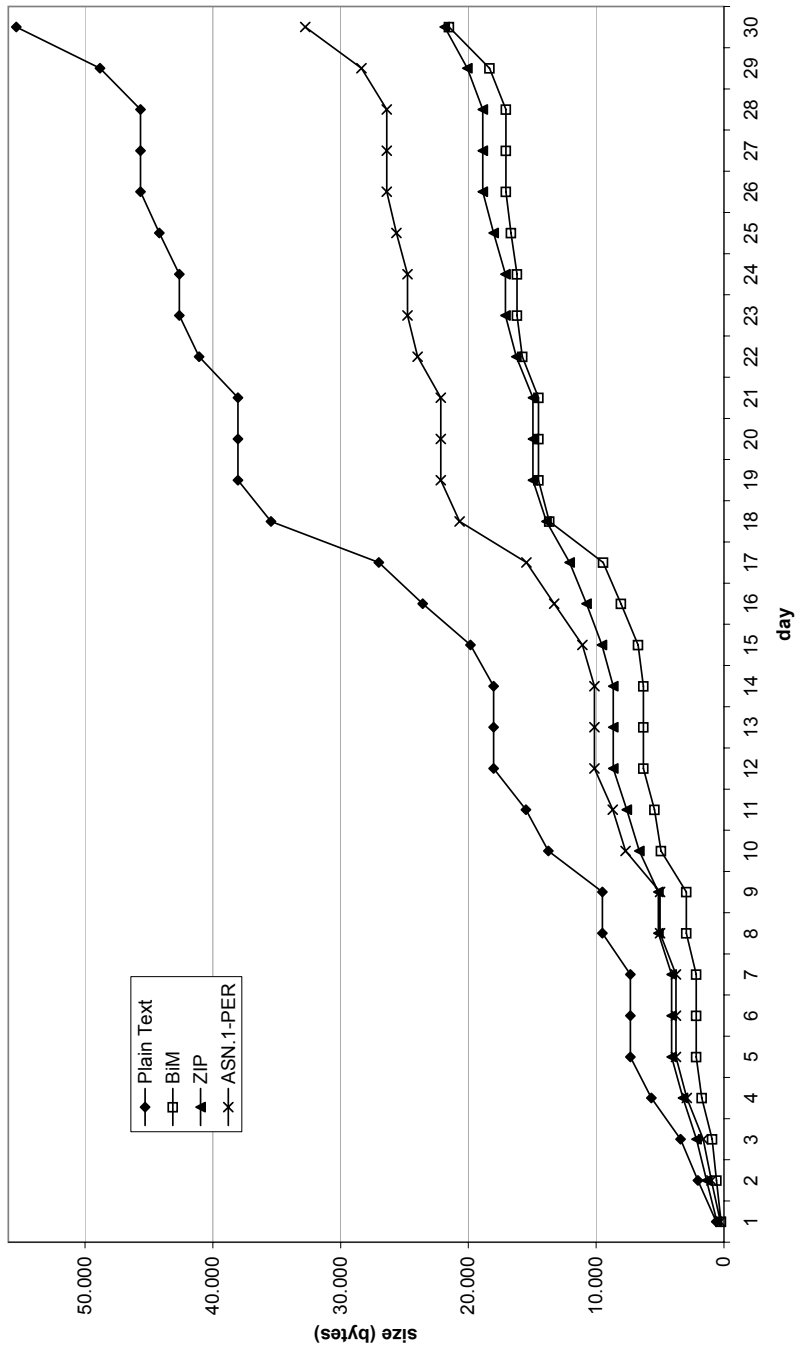


Figure 4.10: Cumulated byte size for Use Case 2: daily retrieval of the RSS feed – Update mode.

Table 4.8: Cumulated byte size for Use Case 2.

	Full		Update	
	size (bytes)	reduction (%)	size (bytes)	reduction (%)
Plain text	524,604		55,382	89.40
ZIP	110,885	78.86	21,858	95.83
ASN.1-PER	352,100	32.88	32,775	93.75
BiM	104,993	79.99	21,512	95.90

native serialization for different use cases. Furthermore, this task force has recently released a new working draft describing measurement aspects, methods, caveats, test data, and test scenarios for evaluating the potential benefits of an alternative serialization for XML [100]. Other W3C activities related to alternative XML serialization include the efficient transportation of non-XML-based data within XML-based data only, e.g., *XML-binary Optimized Packing* [101], and *Resource Representation SOAP Header Block* [102]. Finally, W3C supports in [103] the *WAP Binary XML* (WBXML) as an alternative serialization technique for *Wireless Markup Language* (WML) documents. WML is an XML-based language intended as markup language and is mostly used on WAP-enabled cell phones.

The Web service community is currently developing alternative XML serialization schemes known as *Fast Infoset* [104] and *Fast Web Services* [105]. The latter is built upon ASN.1; the former uses an indexing mechanism that associates an index to each XML element, enabling its usage for further occurrences of the same XML element, i.e., highly repetitive content will benefit from this approach. However, for small and complex XML documents the index table is again a burden. Performance results comparing these two approaches with other binary XML encoding schemes are not yet available.

Finally, we want to mention two proprietary solutions, namely XMill [106] and XMLPPM [107]. The former exploits the self describing nature of XML for compression by leveraging existing compression algorithms and tools like ZIP compression and some simple data type specific compressors. The latter is a compression tool for XML documents that combines the *Prediction by Partial Match* (PPM) and the *Multiplexed Hierarchical Modeling* algorithms. Currently these alternative XML serial-

ization solutions are not yet stable for real-life applications, but progress is being made. In [108] a comparison of XMill and XMLPPM to other alternative serialization techniques is made in terms of compression efficiency. The results demonstrate that BiM is superior to these solutions.

4.7 Conclusions and Original Contributions

As XML is more and more being used in multimedia applications to represent miscellaneous information, the verbosity of the XML format is becoming a concern, especially in constrained environments. In this chapter, we studied possible solutions to this problem using alternative XML serialization formats and by supporting an XML update functionality so only the modified information is transmitted.

First, we investigated the prerequisites to create a parser that can handle both textual and non-textual encoded XML data and supports updates. Therefore, the five XML parser models were investigated, namely the Tree Model, the Push Model, the Pull Model, the Cursor Model, and the Mapping Model. This research illustrates that, while the Push and Pull parser models are fast and have the lowest memory requirements, these models are not appropriate to handle XML updates. The Mapping Model proves to be a very good model, however only for pre-defined domain specific applications. Finally, the Cursor Model is to be preferred over the Tree Model because of the inherent XPath support of the model that can be exploited to support XML updates.

As a result, we created a parser according to the Cursor Model that is capable of handling textual and non-textual encoded XML data. Applications can use this parser to handle XML data without being aware of the actual content encoding format. Indeed, the usage of this parser enables transparent access to XML-based data by shielding users, i.e., application developers, from its encoding format.

Next, we studied potential alternative XML serialization formats, namely ZIP, ASN.1-PER, and BiM, and evaluated them against plain-text serialization for two real-life applications: a UMA application where a client device informs a broker service about its usage context and an RSS application. The three alternative serialization types and the traditional textual serialization are used in two modes, namely a full mode that handles complete and valid XML files and an update mode that only processes the differences. Whereas only BiM natively supports up-

dates, our parser was extended to support the update functionality for the other serialization techniques by application domain-specific extensions.

The results of the evaluation show that BiM serialization is the best solution as alternative XML serialization format in terms of overhead reduction efficiency as it reduces the required bandwidth and the associated costs by more than 92% for the first use case and nearly 96% for the second use case. This is mainly achieved thanks to the native support of updates. Furthermore, BiM-serialized data can be processed in the binary domain. These characteristics make BiM a good alternative serialization type with regard to compression efficiency and usability.

However, the tests clearly demonstrate the extremely slow parsing and creation of BiM data. This is partially explained due to the usage of the MPEG reference software implementation of MPEG-B, which is not optimized for speed. It is expected that (commercial or open source) implementations will exhibit a significantly better runtime behavior because the MPEG-B BiM specification does not have any inherent limitation that would prevent this. Currently, however, there are no such implementations available. Hence, BiM is currently not a recommended solution, as long as optimized encoders and decoders are not available. This is especially true for constrained devices as the additional processing time, and hence power consumption, nullifies BiM's advantages with regard to compression efficiency. Unfortunately, we were not able to create an optimized BiM encoder and decoder ourselves.

A practical alternative serialization method is the ZIP compression strategy. Its main advantage is that most end-user devices already have built-in support for ZIP compression and decompression. Furthermore, it is extremely fast and its compression ratio is only slightly worse than BiM encoding, especially if the XML data contains many string data types as in Use Case 2. Its main disadvantage is the fact that it does not natively support updates. This can only be achieved by a proprietary solution.

The lack of support for updates is also the main disadvantage for the ASN.1-PER method. On top, ASN.1-PER requires an encoder and decoder specifically constructed for the data it has to process, i.e., every XML Schema needs its own software module. Although this encoder and decoder can be created automatically, it prevents the construction of a generic parser. On the other hand, ASN.1-PER has acceptable execution speed and very high compression ratio if no string data types

are present. This makes ASN.1-PER an acceptable solution for specific applications.

Alternative XML serialization formats can address the verboseness and the non-existing update capabilities of XML trees. By letting XML parsers handle these additional serialization formats, the complexity of handling XML is not increased for the application developers. In fact, they do not need to be aware of the actual content encoding format. End users with mobile devices may benefit from such alternative serialization formats by means of paying less for actually sending or retrieving the same data.

The research that has led to this chapter of this thesis is also discussed in our following publications and MPEG contributions.

1. Christian Timmerer, Stephen Davis, Itaru Kaneko, Spencer Cheng, and Robbie De Sutter. Report of CE on MPEG-21 Binarisations. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10974, Redmond, Washington, USA, July 2004

2. Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Using MPEG-21 Part 16 in Applications. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11325, Palma De Mallorca, Spain, October 2004

3. Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Evaluation of Models for Parsing Binary Encoded XML-based Metadata. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, pages 419–424, Seoul, Korea, November 2004

4. Christian Timmerer and Robbie De Sutter. CE Report on MPEG-21 Binarization. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11744, Hong Kong, China, January 2005

5. Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Multimedia Metadata Processing: a Format Independent Approach. In *Proceedings of the 9th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 343–348, Grindelwald, Switzerland, February 2005

6. Christian Timmerer, Ingo Kofler, Johannes Liegl, Hermann Hellwagner, Robbie De Sutter, Wim Van Lancker, and Rik Van de

Walle. Report of CE on MPEG-21 Binary Format. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11858, Busan, Korea, April 2005

7. Robbie De Sutter, Sam Lerouge, Davy De Schrijver, and Rik Van de Walle. Enhancing RSS Feeds: Eliminating Overhead through Binary Encoding. In *Proceedings of the IEEE 3rd International Conference on Information Technology and Applications*, pages 520–525, Sydney, Australia, July 2005

8. Wim Van Lancker, Robbie De Sutter, Davy De Schrijver, and Rik Van de Walle. A Framework for Transformations of XML within the Binary Domain. In *Proceedings of the 10th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 29–34, Innsbruck, Austria, February 2006

9. Robbie De Sutter and Rik Van de Walle. Saving Bandwidth for RSS Feeds by using ASN.1 in E-learning Applications. In *The 9th IASTED International Conference on Computers and Advanced Technology in Education*, Lima, Peru, October 2006. Accepted for Publication

10. Robbie De Sutter, Sam Lerouge, Peter De Neve, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Comparison of XML Serializations: Cost Benefit vs. Complexity. *Multimedia Systems*. To appear (DOI : 10.1007/s00530-006-0044-y)

Chapter 5

Video Scalability

5.1 Introduction

In this chapter, an important part to enable Universal Multimedia Access is discussed, namely the multimedia (in particular audio-visual) content itself. UMA's objective is to supply any possible end-user device with content from one *single* content base. This implies that from the original content, multiple versions must be derived that are consumable on a wide range of devices, from low-end terminals, such as PDAs, up to high-end terminals, such as *High Definition Televisions* (HDTVs). These devices not only differ in display size, but also in processing capacity, supported decoders, (network) connectivity, and so on. Note that multiple copies of the content base may be distributed over multiple (geographically disperse) servers for load balancing and load optimization reasons.

In Chapter 3, we introduced the concept of a *content adaptation engine*. This engine modifies audio-visual content so it becomes usable for a given context. In this chapter, we investigate the techniques how to perform this content adaptation.

One possible technique is to *transcode* the content [109]. Classical transcoding is modifying the original content from one format of encoding into another format, usually by decoding the content and then re-encoding it compliant to the desired type. This is a time-consuming step due to the decoding and encoding steps. More recent transcoding algorithms avoid these steps and work in the compressed domain. As

a result, they are faster and also capable of changing particular characteristics of the content, such as changing the frame rate and the bit rate.

Another technique to achieve the UMA goal of supplying any kind of device with consumable content from a single content base is by encoding it in a *scalable* way. As such, it becomes feasible to derive different versions from a single content base by simple, mostly truncation, operations on the bitstream.

Within this chapter, we elaborate on the different types of scalability for video content and give an overview of different scalable video coding techniques. We also discuss a fast object tracking system. This system consists of a collection of algorithms with low time complexity making it possible to automatically track moving objects. Because our system reuses information from the encoder, the object tracking can be performed in real-time. The algorithms are implemented in an MPEG-4 *Fine-Granularity Scalability* (FGS) encoder and enhance its *Region-of-Interest* (ROI) functionality.

5.2 Types of Video Scalability

Video Scalability means the possibility to manipulate a video stream in the compressed domain so a new video stream can be obtained with different characteristics. There are three main types of video scalability:

- temporal scalability.
- signal-to-noise ratio scalability.
- spatial scalability.

A video can be scaled in order to reduce the required bandwidth or to lower the necessary processing power. The downside is degradation in (perceived) quality.

5.2.1 Temporal Scalability

Temporal scalability modifies the number of *frames per second*. The illusion of motion in a (natural) video is created by projecting roughly 24

slightly different images per second to the human visual system. Using more images per second enforces this illusion so a more fluent motion is perceived; using fewer images per second introduces a jerkiness that humans can observe.

Temporal scalability is the easiest form of scalability and can be realized by almost all existing video coders. If the video encoder does not eliminate the temporal redundancy between frames (e.g., Motion JPEG2000 [110]), temporal scalability is accomplished in a trivial way by not transmitting or not decoding all the frames of the video stream but only, for example, the odd frames in case of a dyadic temporal reduction.

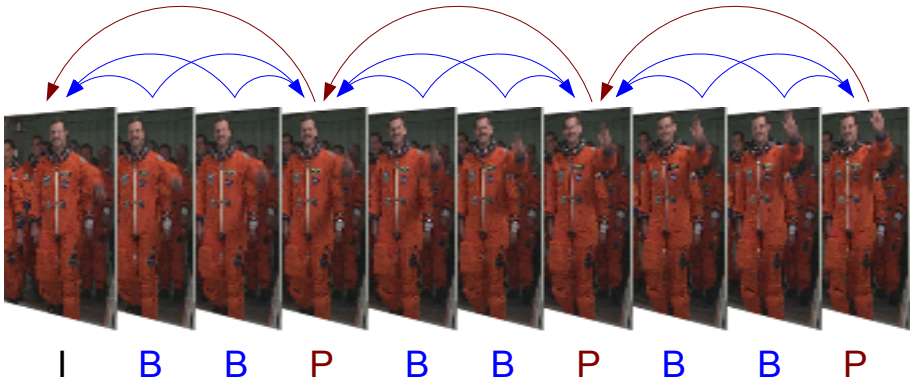


Figure 5.1: Ten successive frames from the "Crew" sequence with IBBPBBPBBP-GOP structure. The P-frames depend on the previous I or P-frame; the B-frames depend on the previous and next I or P-frames; the I-frames are independent of any other frame.

However, most video encoders try to minimize *inter-frame redundancy* – i.e., removing the similarities between successive frames – because this results in a higher compression ratio [111]. These kinds of encoders require a different strategy to allow temporal scalability. Typically, video encoders do not work on complete video streams, but on a smaller *group of pictures* (GOP). A GOP is the base work-unit of the video encoder, hence the inter-frame redundancy within a GOP is to be eliminated. A GOP starts with a frame that is encoded independently from any other frame, also called an *intra-coded* frame or I-frame. The remaining frames of the GOP are *inter-coded* frames. This means that these frames use information from previous frames (also called P-frames) or from previous and future frames (also called B-frames). Figure 5.1 illus-

trates this concept for a GOP of ten frames. Because some frames are referenced by other frames, these reference frames cannot be eliminated when executing a frame rate reduction. Indeed, temporal scalability can be performed by first dropping the unreferenced B-frames. If necessary, the P-frames can be dropped back-to-front.

It should also be mentioned that recent video coding technologies (such as the MPEG and ITU.T H.264/MPEG-4 Part 10 – Advanced Video Coding and Scalable Video Coding standards) refine these principles. They allow GOP structures that do not start with an intra-coded frame; GOP structures that are changed during encoding; frames that are partitioned in several slices of different types (intra-coded I-slices and inter-coded P- or B-slices); P-slices that use information from subsequent slices, et cetera.

5.2.2 Signal-to-Noise Ratio Scalability

Signal-to-Noise Ratio (SNR) scalability modifies the number of visual artifacts in and sharpness of the video stream, as illustrated in Figure 5.2.

(a) original image (b) overall quality reduction (c) selective quality reduction

Figure 5.2: Examples of SNR Scalability.

SNR scalability can be accomplished by several techniques. In case of classic *Discrete Cosine Transform* (DCT) based video coders, the easiest way is by only using the upper-left coefficient(s) of the DCT-

matrix. It can also be realized by creating a multilayer version of the video content, as illustrated in Figure 5.3, so the base layer contains the coarsest version and the higher layers enhance this version. For wavelet-based video coders, SNR scalability can be achieved by processing only the LL-subband as will be explained in Section 5.3.3.

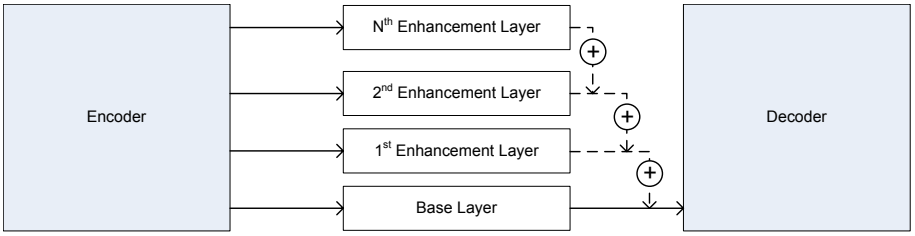


Figure 5.3: Schematic representation of a multilayer video encoder / decoder.

5.2.3 Spatial Scalability

Spatial scalability modifies the number of pixels in the video content, resulting in a change of resolution. This form of scalability is very important to support the many different display sizes of all existing terminals. Indeed, it is a waste of resources to stream and decode a high definition video with a resolution of 1920×1080 pixels on a *Common Intermediate Format* (CIF) sized (352×288 pixels) display. Figure 5.4 gives an overview of the contemporary display resolutions.

Spatial scalability can be accomplished by resizing or by cropping the original video content. Either way, it is the most difficult form of scalability to realize with traditional DCT-based video coders and is only feasible with an optimized video encoder. Encoders supporting this form of scalability tend to use a multilayer approach, as illustrated above in Figure 5.3. Here, the base layer contains the video feed in the lowest resolution, for example *Quarter CIF* (QCIF). By adding the information of one or more layers, a higher resolution can be obtained. As such, spatial scalability can be accomplished by only transmitting or decoding those layers required to obtain the desired resolution. When using a wavelet-based video coder, spatial scalability can be obtained more easily as will be explained in Section 5.3.3.

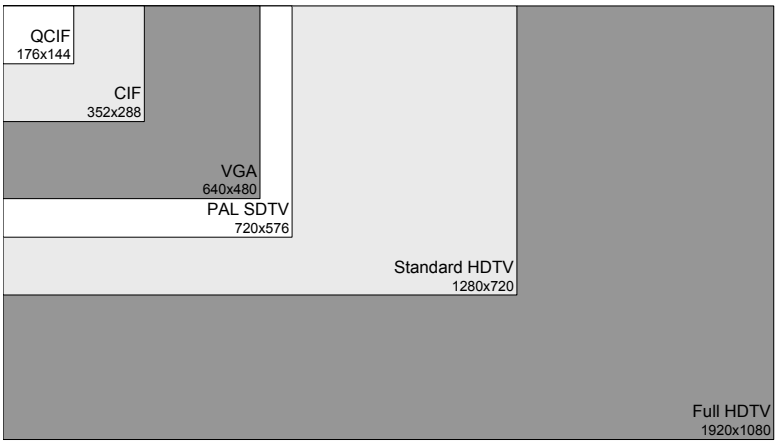


Figure 5.4: Common display resolutions.

5.3 Scalable Video Coders

In this section, we give an overview of recent video coding technologies that support scalability. As temporal scalability can easily be realized as discussed in Section 5.2.1, we will only discuss recent video standards that have specific support for at least one of the two other types of scalability, namely SNR scalability or spatial scalability.

5.3.1 Fine-Granularity Scalability

The MPEG-4 Fine-Granularity Scalability standard [112] is a scalable extension of the MPEG-4 Visual standard [113]. Its goal is to allow streaming servers to adapt the bit rate of streamed video to the characteristics of the network and, by extension, to the possibilities of the end-user device. Basically, FGS creates two video layers (see Figure 5.5): a base layer that contains a low quality version of the video that can be streamed and decoded under any circumstances by any FGS-compliant device, and an enhancement layer that improves the quality of the base layer video. The enhancement layer is constructed in such a way that the streaming server can truncate this bitstream at any desired bit location, resulting in a loss of visual quality, but empowering the streaming server to comply with the targeted bit rate. Hence, FGS supports SNR scalability for a given video stream.

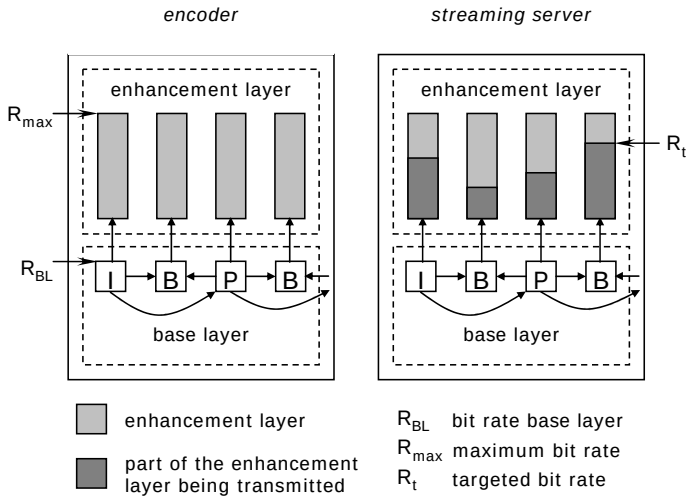


Figure 5.5: Real-time bit rate adaptation in MPEG-4 FGS. The streaming server will only send the dark gray parts of the enhancement layer (the base layer is always sent) to comply with the targeted bit rate [114, 115].

The base layer is traditionally encoded with MPEG-4 Visual Simple Profile [113, 116]. However, other encoding schemes for the base layer are suggested in [117–120]. As such, FGS can be seen as an enhancement scheme on top of existing coders delivering additional quality and enabling various additional features.

When referring to FGS encoding, we actually refer to the encoding of the enhancement layer, as the base layer is encoded by another codec. For the encoding of the FGS enhancement layer, different techniques are possible: Embedded Zero-tree Wavelet encoding [121, 122], matching pursuit coding of the image residue [123], and bit-plane DCT residue encoding [124] to name a few. Within MPEG, the latter was chosen as the encoding technique for the MPEG-4 FGS enhancement layer.

The enhancement layer receives as input the DCT residue values from a *macroblock* in a VOP[1], i.e., the values obtained after subtracting the de-quantized DCT coefficients of the base layer from the original DCT coefficients (Figure 5.6a). The resulting residue matrix inherits all characteristics of a DCT matrix. The difference to more traditional video encoding schemes is a novel approach to encode these residual values.

[1]MPEG-4 defines a *Video Object Plane* (VOP) as a time sample of a video object, such as a video frame.

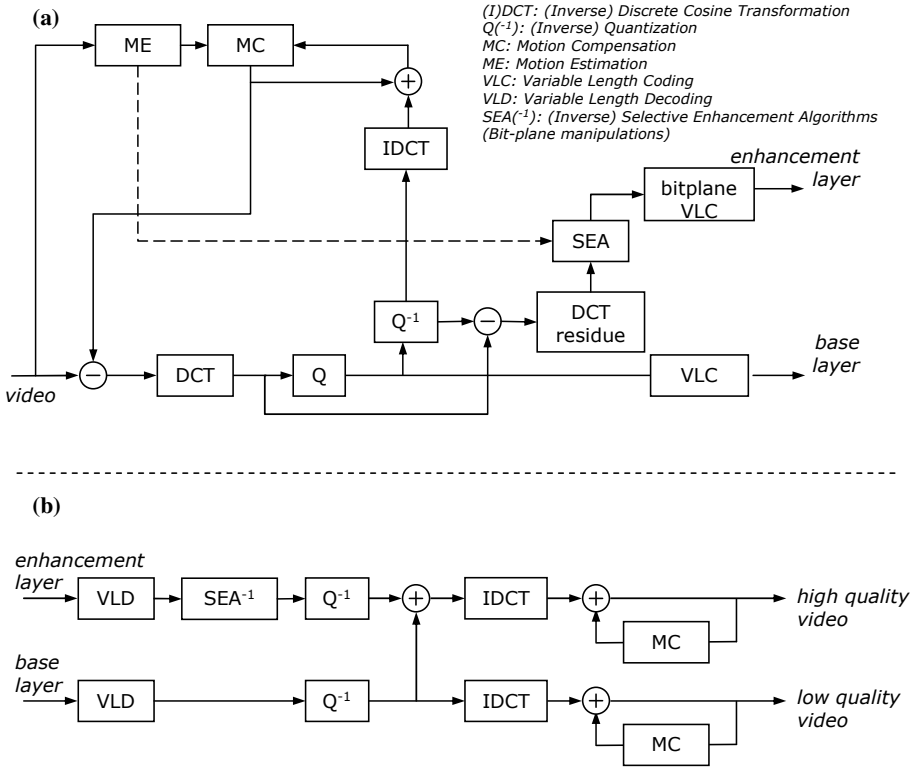
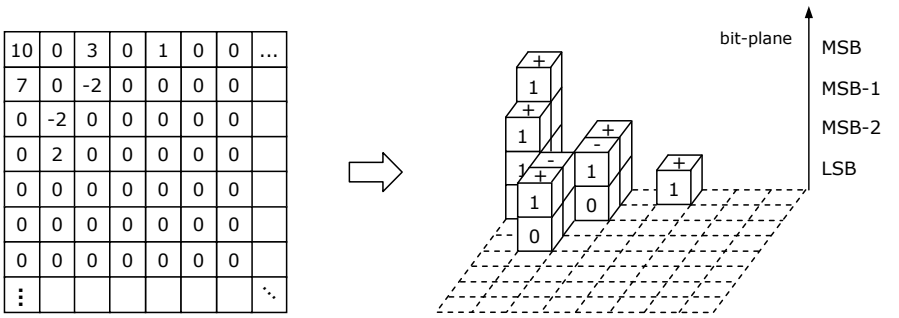


Figure 5.6: (a) MPEG-4 FGS encoder and (b) MPEG-4 FGS decoder with bit-plane shifting logic.

Most traditional video encoding schemes apply run-level encoding on the values of the quantized DCT matrix of the macroblock obtained after a specific scan method, such as raster-scan or zig-zag scan. The bit-plane DCT residue encoding is performed by zig-zag scanning the values of the residue matrix and by placing them in their binary form as a column in a matrix (Figure 5.7). The sign of a value is stored separately, so only the absolute value is used for the binary representation. A *bit-plane* is one row in this matrix, thus a sequence of 256 bits in case the size of a macroblock is 16×16 pixels, as in MPEG-4 Visual Simple Profile. The top bit-plane is the *Most Significant Bit-plane* (MSB); the lowest bit-plane is called the *Least Significant Bit-plane* (LSB).

The encoder processes the binary matrix representation of the residue DCT values bit-plane by bit-plane, starting with the MSB and ending with the LSB. Each bit-plane is translated to unique symbols and en-



MSB	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0	0
MSB-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0	0
MSB-2	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0	...	0	0
LSB	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	...	0	0

Figure 5.7: Bit-plane representation of a DCT residue matrix.

coded by the run-length *variable length coding* (VLC) technique.

The encoder or a streaming server can drop one or more bit-planes – starting with the LSB up to the MSB. *Dropping* a bit-plane means not completely encoding or not completely transmitting the bit-plane and, as a result, using fewer bits for the enhancement layer. This technique enables the encoder or the streaming server to comply with the targeted bit rate R_t in Figure 5.5. If one or more bit-planes are (partially) dropped, the reconstructed values at the decoder side are less precise. These less accurate values can still be used to rebuild the macroblocks of the VOP, but with a decreased visual quality. When dropping bit-planes, large residue values are more likely to have a reconstructed value than small residue values as the larger values will have bits in the upper, non-discarded, bit-planes.

Finally, we briefly introduce the support for Region-of-Interest in MPEG-4 FGS. A ROI is an area within the video scene that is marked as more important than the remaining areas. For example, a person in a surveillance video is a more important object than the background. Coders supporting the ROI concept create a video stream that ensures that the visual artifacts first occur outside of the selected region. As such, this is a special form of SNR scalability so only selected parts are not degraded (Figure 5.2(c)).

MPEG-4 FGS natively supports the ROI concept. It is done by executing an additional operation on the binary matrix representation before bit-plane dropping and works on the complete DCT residue matrix of a macroblock. A *shift* operation is performed on its values, i.e., multiplying the matrix coefficient values with 2^{α} (α is the number of bit-planes). Because the binary representation of these shifted values appears in higher bit-planes, they are placed more in front during encoding. This improves the probability they will be present in the received – and probably truncated – enhancement layer. Hence, the probability increases for all values within this DCT residue matrix to be reconstructed with a higher precision. As a result, this operation shifts selected and complete macroblocks by a given number of bit-plane levels and therefore can be used to support ROI. Obviously, the adequate information about the shifting is added for each macroblock to the resulting bitstream so the decoder can perform the inverse transformation. This is represented by the $\texttt{SEA}^{-1}$ block in Figure 5.6b.

5.3.2 Scalable Video Coding

After MPEG finalized its MPEG-4 Visual standard, it teamed up again with the ITU.T *Video Coding Experts Group* (VCEG) in the *Joint Video Team* (JVT) that previously led to the highly successful MPEG-2 standard (dubbed H.262 by ITU.T). The JVT recently finalized the *H.264/MPEG-4 Part 10 – Advanced Video Coding* (AVC) standard. This coding standard achieves much higher compression ratios while maintaining the same video quality levels when compared to other video coders [125].

The AVC standard does not natively support scalability (temporal scalability excluded). As this was perceived as a missing feature, the JVT group started developing the *Scalable Video Coding* (SVC) standard, which will be amendment 3 to the AVC specification [126, 127]. SVC is a video coding standard that natively supports SNR and spatial scalability, the latter being the main difference with FGS. After the initial tests and evaluations of the different proposals, AVC was selected as the base video coder for SVC.

In the remainder of the section, we explain how the different types of scalability are achieved. Since the SVC standard is still work in progress, this information is preliminary and might change. More information can found in [25, 128].

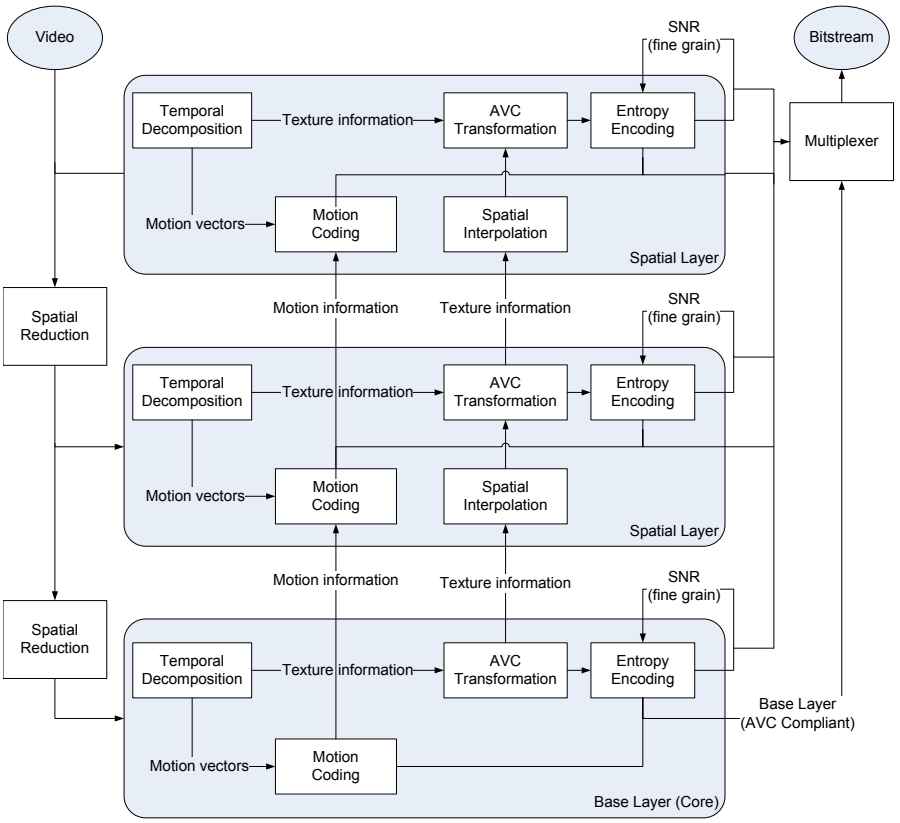


Figure 5.8: SVC encoder with three levels of spatial scalability and support for fine-grain SNR scalability [127].

To obtain temporal scalability, SVC exploits the AVC capabilities by creating a hierarchical, pyramid-like structure of B-pictures.

Spatial scalability is achieved by creating a multilayer video encoding structure – Figure 5.8 shows for example a three-layered structure. The base layer contains the lowest resolution of the video stream. When adding the enhancement layers, the resolution of video is enlarged. Practically, the input video is first scaled by the *"Spatial Reduction"* block(s) to the desired video resolution. The spatial scaling factor is not limited to two (to get half resolution), but any factor may be used and also cropping of the video is allowed.

Next, the video with the lowest resolution is the input for the base layer. This video is encoded by a regular AVC encoder, as such an

AVC-compliant bitstream is sent to the multiplexer. The encoding of the enhancement layers is done slightly different as these layers not only receive the original input video scaled to the desired resolution, but also the decoded and up-scaled version of the video of the layer below the current layer. Furthermore, the layer receives information about the motion estimation from the layer below. This inter-layer information exchange is an important novel concept for multilayer video coding and allows the encoder to improve the motion estimation for the current layer, for example, by accelerating the algorithm.

Finally, these layers are multiplexed into one bitstream. The bitstream is constructed so at least the base layer can be decoded by any AVC-compliant decoder. Even more, active network nodes can be created that are capable of removing one or more enhancement layers. Hence, networks can scale the streams to fit their bandwidth capacity.

SNR scalability in SVC is achieved in two ways: a *fine-grain* solution, inspired by the FGS technology, and a *coarse-grain* system, using additional layers.

The fine-grain solution creates SNR scalability within a layer using a method known as *progressive refinement*. This repeats the quantization step in the entropy encoder several times, see also Figure 5.8. During the first execution of this encoding step, a large quantization step size is used. This results in a coarse version. In subsequent executions the quantization step is reduced, resulting in a more accurate image. Only the difference between this enhanced image and the image obtained during the previous step is stored, as such reducing the overhead.

The coarse-grain solution provides SNR scalability by adding enhancement layers to the multilayer structure. The same concept as for spatial scalability intended layers is used for the SNR scalability intended layers. The difference is that the latter works with the same video resolution as the layer below, but the quantization step is lower. In other words, the resulting stream of this layer is a more accurate version compared to the resulting stream of the layer below. Again, inter-layer communication is used to improve the processing. Note, that mixing layers intended for spatial scalability with layers for coarse-grain SNR scalability is possible.

5.3.3 Wavelets

While wavelet-based (video) coders are not yet common due to their high complexity, we briefly discuss this technology because of its good scalability features.

A wavelet-based transformation is seen as an improved alternative for DCT as it is insensitive to the introduction of *block artifacts*. Block artifacts appear because DCT works on blocks of, for example, 16×16 pixels and not on a complete video frame directly. During the decoding and after the inverse DCT, the blocks are tiled to reconstruct the video frame. However, as the encoding and decoding steps are lossy transformations, these blocks differ, amongst other things, in color intensity. This difference is most visible at their borders, hence the blocky appearance.

Wavelets do not work on chunks of the frame, but on the frame as a whole – although for optimized parallel processing the image is sometimes split-up. By applying a wavelet filter in the horizontal and vertical direction, such as the Haar filter, four *subbands* are obtained. These subbands are notated as LL, LH, HL and HH. LL is an approximation of the original image, the other subbands contain information about the difference between the LL image and the original image. The decomposition can be repeated several times using the LL subband as input image. Figure 5.9 shows a wavelet decomposition of the image "Lena."

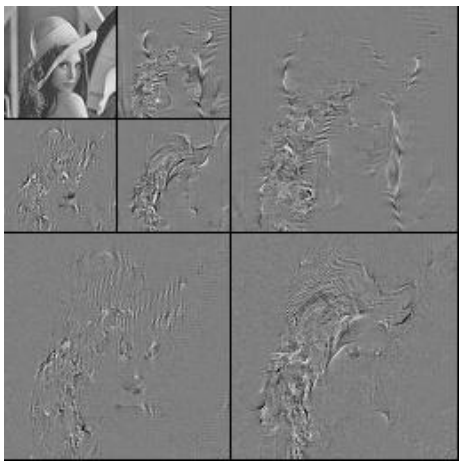


Figure 5.9: Wavelet decomposition of the image "Lena."

The image in the LL subband is not only a coarser version of the original image, it is also half the size thereof. In other words, if only the LL subband of a wavelet decomposed image is decoded, the resulting image is a SNR and spatial scaled version. By combining particular subbands, it is possible to obtain a pure SNR scaled version, a pure spatial scaled version, or a combination of both.

The wavelet-based coding system as described above, can not only be used for still images, but also for video sequences. Indeed, one can simply code every frame independently (e.g., Motion JPEG2000 [110]). However, these methods do not optimally remove the temporal redundancies between frames. To realize the latter, many advanced *motion-compensated temporal filtering* (MCTF) methods are described in the literature, for example, the *motion-compensated three-dimensional subband/wavelet coding* (MC-3DSBC) [129] and the *motion-compensated embedded zero-block coding* (MC-EZBC) [130, 131] to name a few.

5.4 Object Tracking

As explained earlier, a Region-of-Interest is a particular area in a video scene that is seen as more important than the surrounding areas as illustrated in Figure 5.2(c). Usually, this implies that one or more objects in this region deserve particular attention. An issue that arises is the identification of these important regions in the video scene and the tracking of the object(s) throughout the video sequence. The remainder of this chapter focuses on the latter issue and introduces novel algorithms to make the pursuing of selected objects possible, also called *object tracking*. These new algorithms reuse the motion vectors that are estimated during the motion estimation step of the video encoder. While other comparable object tracking algorithms work in the uncompressed domain and have a very high computational complexity, our algorithms work during the actual encoding of the video in the compressed domain and use the information directly available from the encoder itself. This results in an object tracking scheme that is very fast and that can be used for real-time streaming applications.

Note, our techniques do not identify the objects, also called *object segmentation* or *object classification*. It is assumed that the object that has to be followed is determined beforehand, either by manual selection (e.g., by a human operator) or automatically by using a segmentation

technique. An overview of these techniques is given at the end of this chapter.

Our algorithms are implemented on top of the MPEG-4 FGS reference software[2]. We enhance the ROI capabilities of this coder by automatically moving or resizing the ROI according to the motion or the change in size of the tracked object. Pseudo-code listings as an illustration of our algorithms are given in the Appendix B.

5.4.1 Fast Object Tracking Techniques

During the encoding of the base layer, with an MPEG-4 Visual Simple Profile encoder, the *motion vector field* is estimated. This is a mathematical representation of the displacement of a group of pixels between related (previous) frames of a video stream. For each *macroblock*, one or more *motion vectors* are estimated. We reuse this information to determine the motion of the object within the ROI, represented by a *object motion vector*. This is illustrated by the dashed arrow in Figure 5.6a. The object motion vector is used to move the location of the ROI and, as such, to follow the motion of the object. Furthermore, as the object becomes larger or smaller, for example due to camera zooming, the ROI must grow or shrink accordingly.

The novel algorithms discussed in this section work on a frame[3] per frame basis; for each frame the following steps are executed (pseudo-code in Listing B.1):

- First, the macroblocks that are part of the ROI are identified.
- Hereafter, the motion of the object within the ROI is determined, using the motion vectors from selected macroblocks. This results in the translation of the ROI.
- Next, the ROI is resized automatically to cope with the object resizing using information from the motion vector field.
- Finally, the relevant macroblocks of the updated ROI are identified and these are shifted by the shift value α.

[2]The reference software source code is available at `http://megaera.ee.nctu.edu.tw/mpeg`. We used the *Microsoft-FDAM1-v2.4-030305* version.

[3]Because our algorithms are independent of the actual video encoding technology used, we prefer to use the generic term "frame" instead of "VOP."

To end this section, a *cloaking* technology is introduced and the time complexity of the algorithms is given.

Selecting the macroblocks

In order to support the ROI concept, the default MPEG-4 FGS method uses a matrix that contains the shift values for the macroblocks. This matrix, also called the *ROI mask*, creates a direct mapping between a macroblock and the Region-of-Interest. Indeed, if the shift value of a macroblock is larger than zero, shifting of this block will occur, hence the macroblock is part of the ROI. On the other hand, if the shift value of a macroblock is zero, no shifting will occur, hence the macroblock is no part of the ROI.

Because an MPEG-4 Visual Simple Profile base layer encoder uses macroblocks of 16×16 pixels, these rather large blocks of pixels are not suitable for fine-detailed object tracking. Hence, we do not use this default ROI mask, but we create an *enhanced ROI mask* that works as a separate floating layer on top of the frame and the macroblocks within the frame. This enhanced mask no longer needs to be aligned with the boundaries of the macroblocks. Hence, our algorithms are independent of the size of a macroblock, which is determined by the video encoding specification. Figure 5.10 illustrates the enhanced ROI mask concept as a floating layer (represented by the black grid) on top of the macroblocks (represented by the white grid).

In addition, the enhanced mask is approximately the size of the object that is being traced, not the size of the frame. More precisely, the enhanced ROI mask usually has the same size of the region that is visually improved and a *cloaking* mechanism ensures that the size of the visually improved region and the size of the tracked object do not need to coincide. Nevertheless, we first explain the object tracking system without the cloaking technique.

The enhanced ROI mask is represented by the matrix T with dimensions $T_C \times T_R$; T_C is the number of columns and T_R is the number of rows in the enhanced ROI mask. The value of an element in matrix T – referred to as $T(x, y)$ – represents the shift value α that has to be applied to the corresponding macroblock(s). Note that if $T(x, y) = 0$, no shifting will occur, allowing to create an arbitrary-shaped ROI that holds arbitrary-shaped objects. Further, an element $T(x, y)$ can represent any block

size; here we use 8×8 pixels as block size, which is a quarter of the default 16×16 pixels macroblock size in MPEG-4 FGS. This creates a more fine-grained ROI mask, allowing a more neatly-fitting selection of the object. The discussed algorithms can easily be modified to cope with any other size. For the remainder of this chapter, we will assume 8×8 pixels as block size for the ROI mask elements and a macroblock size of 16×16 pixels.

As the enhanced ROI mask floats on top of the frame, its location is also required. This is specified in pixel coordinates (denoted as L_X and L_Y) and locates the upper-left corner of the mask. Note, this location does not need to be aligned to a macroblock boundary.

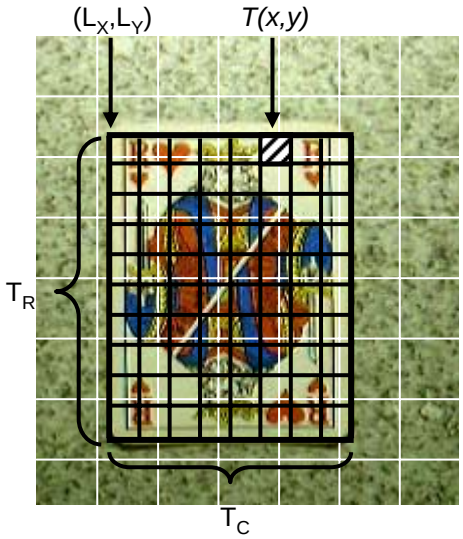


Figure 5.10: A part of a frame where the card is selected as the ROI. The enhanced ROI mask is a floating layer on top of the frame (represented by the black grid) and it is not aligned to the boundaries of the macroblock (represented by the white grid). The enhanced ROI mask has dimensions $T_C \times T_R = 8 \times 10$. An element of the mask is notated as $T(x, y)$. The upper-left corner of the mask has the coordinates (L_X, L_Y).

Because of the decoupling of the ROI mask with the macroblocks, a mapping algorithm is required to determine the macroblocks that are (partially) covered by the ROI. Each element $T(x, y)$ in the enhanced ROI mask matrix T covers at least one macroblock, denoted as $m_{i,j}$.

The indices i and j are calculated by the formulas:

$$i = \left\lfloor \frac{L_X + 8x}{16} \right\rfloor, \quad j = \left\lfloor \frac{L_Y + 8y}{16} \right\rfloor$$

where $x = 0, 1, \ldots, (T_C - 1)$, $y = 0, 1, \ldots, (T_R - 1)$, $i \in [0, M[$, and $j \in [0, N[$ (M being the number of macroblocks in a frame horizontally and N being the number of macroblocks in a frame vertically).

The computed macroblock $m_{i,j}$ is tagged as part of the ROI and receives the shift value $T(x, y)$. As $T(x, y)$ represents a block of 8×8 pixels, it is possible that $T(x, y)$ covers other macroblocks as well, as illustrated in Figure 5.11, namely $m_{i+1,j}$, $m_{i,j+1}$, and $m_{i+1,j+1}$. In Figure 5.11(b), $T(x, y)$ overlaps with macroblock $m_{i,j}$ and macroblock $m_{i+1,j}$. This can be mathematically expressed by the conditions $(L_X + 8x) \bmod 16 > 8$ and $(L_Y + 8y) \bmod 16 \leq 8$. Other overlaps can be determined in a similar way. All macroblocks that are covered receive the shift value $T(x, y)$, hence in case of Figure 5.11(b) the macroblocks $m_{i,j}$ and $m_{i+1,j}$ receive this shifting value. If a macroblock already received a shifting value, then the highest shifting value is used. In case of Figure 5.11(b), the macroblock $m_{i,j}$ receives the highest shifting value from the ROI mask elements $T(x-2, y-1)$, $T(x-1, y-1)$, $T(x, y-1)$, $T(x-2, y)$, $T(x-1, y)$, $T(x, y)$, $T(x-2, y+1)$, $T(x-1, y+1)$, and $T(x, y+1)$ assuming that $x-2 \geq 0$, $y-1 \geq 0$, and $y+1 < T_C$. Listing B.2 contains pseudo-code to determine the shifting values for all macroblocks.

The time complexity of the described algorithm to determine the macroblocks that are covered by one ROI mask element $T(x, y)$, is $\mathcal{O}(1)$: the calculation of i and j with the given formulas and the determination of overlaps with any neighboring macroblocks is done with a constant time complexity. Note, all multiplication and division operations to select the macroblocks can be executed by (fast) binary bit-shift operations.

Object Motion

Object tracking using the motion vectors of the object is only possible if the motion vector field is available. Traditional base layer encoders, such as implementations of the MPEG-4 Visual Simple Profile specification, do not estimate the motion vector field for intra coded frames. This limitation can be circumvented by different strategies. For example, the encoder can be slightly modified, with minor overhead, in such a way that motion estimation is performed – hence the determination of the

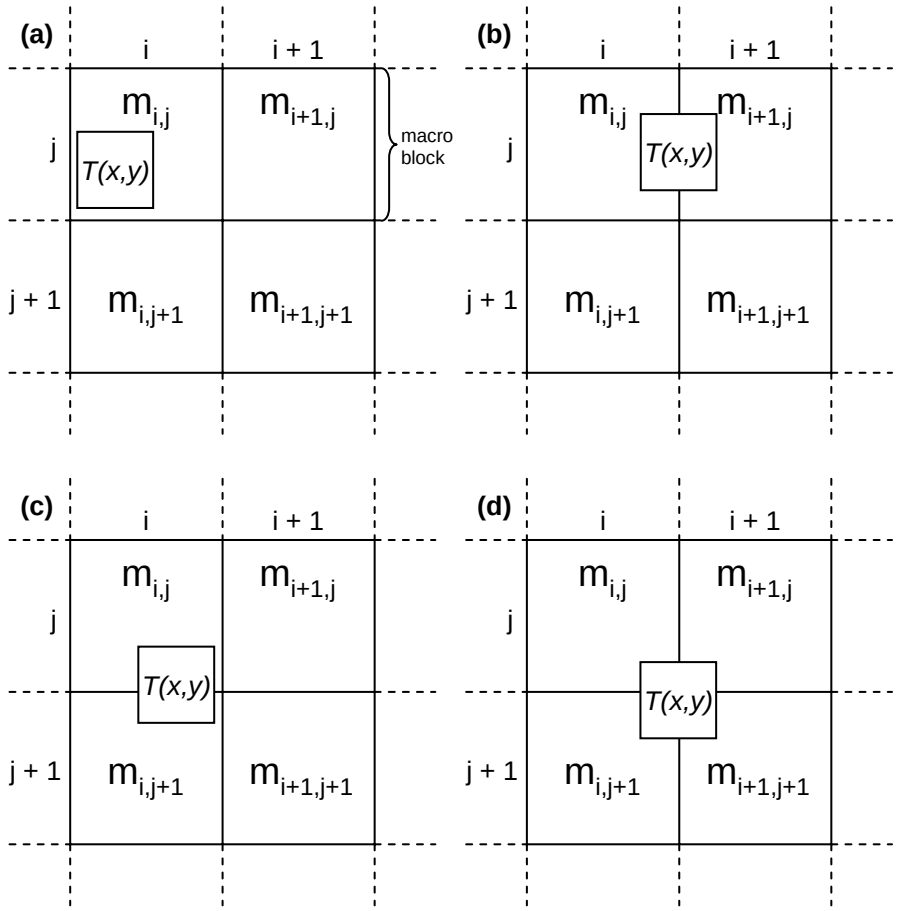


Figure 5.11: Mapping of the ROI mask element $T(x, y)$ onto the macroblock $m_{i,j}$. (a) The ROI mask element (representing 8×8 pixels) fits completely in the macroblock $m_{i,j}$ of 16×16 pixels. (b)(c)(d) The ROI mask element covers neighboring macroblocks as well.

motion vector field – even for intra coded frames. The motion estimation can use the latest available frame as reference. In other words, the dashed arrow in Figure 5.6 always provides the object motion vector field, also for intra coded frames. Note, the motion vector field for intra coded frames does not need to be stored into the resulting bitstream, so this modification of the encoder's implementation does not affect the structure of the resulting encoded video bitstream. Because there is at least one solution available, this and following subsections assume that the motion vector field for the current frame is available.

To determine the motion of the object, the *object motion vector* (`OMV`) is calculated and is used to translate the upper-left corner of the ROI mask resulting in the adjustment of the L_X and L_Y values of the ROI mask. The `OMV` is calculated using the steps explained in the following paragraphs.

For each $T(x, y)$ element, four motion vectors are collected from the motion estimation. Which motion vectors are chosen, depends on the location of $T(x, y)$ as shown in Figure 5.12 (motion estimation is in *4MV mode*[4]). As a result four sub-blocks are selected and the motion vectors thereof are denoted as the vectors $MV^1_{x,y}$ to $MV^4_{x,y}$ (Listing B.5). The selection of the motion vectors is independent of their accuracy; the implementation in MPEG-4 FGS uses half *pixel element* precision as MPEG-4 Visual Simple Profile is used for the base layer.

Next, four *overlapping percentages* (P_1 to P_4) are calculated. They express the overlap of an element $T(x, y)$ with the four selected sub-blocks. In 4MV mode, first the horizontal (d_x) and vertical (d_y) overlap of $T(x, y)$ with the first sub-block is determined. This is illustrated by Figure 5.13 and calculated using the formulas:

$$\begin{aligned} d_x &= 8 - ((L_X + 8x) \bmod 8) &&= 8 - (L_X \bmod 8) \\ d_y &= 8 - ((L_Y + 8y) \bmod 8) &&= 8 - (L_Y \bmod 8) \end{aligned}$$

[4] The 4MV mode divides a macroblock in four 8×8 blocks (*sub-blocks*). The motion vectors of the four sub-blocks are estimated. For the macroblock $m_{i,j}$ these are notated as $MV^{\gamma}_{i,j}, \gamma = 1, 2, 3, 4$. *1MV mode* – where only one motion vector per macroblock is estimated – can also be used by the algorithm.

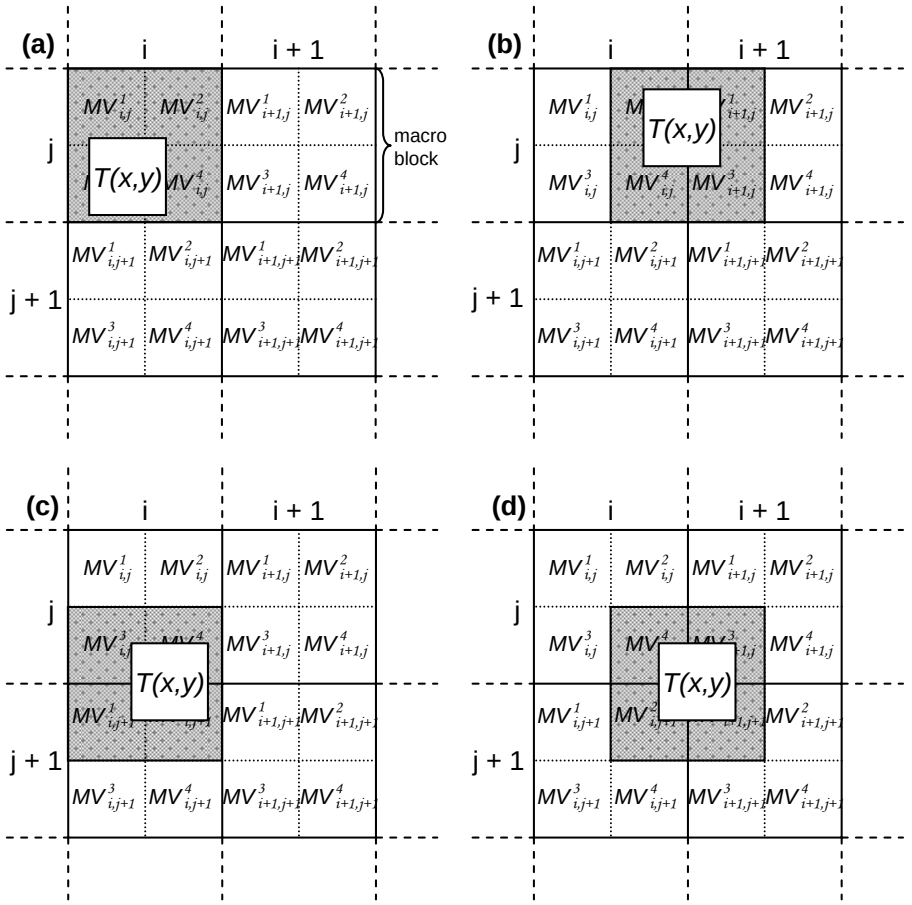


Figure 5.12: Selecting the four appropriate motion vectors for ROI mask element $T(x, y)$, based on the overlap with the macroblocks.

The d_x and d_y values allow the calculation of the overlapping percentages P_γ:

$$\begin{aligned} P_1 &= d_x d_y/64 \\ P_2 &= (8-d_x)d_y/64 \\ P_3 &= d_x(8-d_y)/64 \\ P_4 &= (8-d_x)(8-d_y)/64 \end{aligned}$$

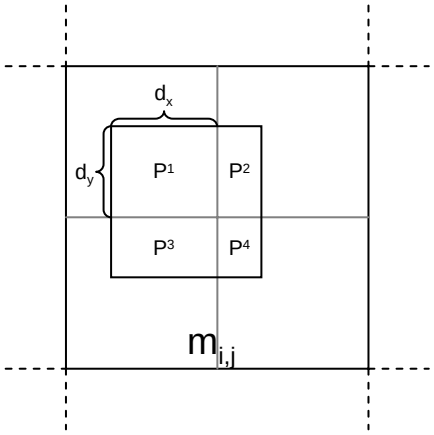


Figure 5.13: A macroblock $m_{i,j}$, divided in four 8×8 sub-blocks, is covered by an $T(x, y)$ element. The figure shows how to determine d_x and d_y and the overlapping percentages P_1 to P_4.

These percentages P_γ are identical for all elements of the ROI mask T, because we use a block size of 8×8 pixels for the ROI mask elements. Pseudo-code on how to calculate d_x, d_y, and P_γ can be found in Listing B.3.

Next, the object motion vector ($\texttt{OMV}_{x,y}$) for the $T(x, y)$ element is determined as the weighted sum of the motion vectors $MV^\gamma_{x,y}$ and the overlapping percentages P_γ by:

$$\texttt{OMV}_{x,y} = \frac{\sum_{\gamma=1}^{4} \left(MV^\gamma_{x,y} \cdot P_\gamma \right)}{\sum_{\gamma=1}^{4} P_\gamma} \tag{5.1}$$

Finally, the OMV is calculated as the average of all $\texttt{OMV}_{x,y}$ vectors:

$$\texttt{OMV} = \frac{\sum_{x=0}^{T_C-1} \sum_{y=0}^{T_R-1} \texttt{OMV}_{x,y}}{T_C T_R} \tag{5.2}$$

The resulting motion vector is the overall object motion vector of the ROI mask (Listing B.4). This vector is used to adjust the L_X and L_Y values.

Formula 5.1 has two additional constraints. The first constraint removes the influence of motion vectors of too small overlapping areas. If P_γ is smaller than a threshold λ, the value of P_γ is substituted by zero. This explains the need of the divisor in Formula 5.1.

The second constraint removes the influence of the motion vectors being part of the border of a ROI mask as these could become less reliable after some iterations of the algorithm. If $T(x, y)$ is part of the border of the ROI mask, the vector $\texttt{OMV}_{x,y}$ is set to the null vector. In this case, the divisor in Formula 5.2 is decreased with the number of ROI mask elements that fulfills constraint (C.2).

These constraints can be expressed as follows:

(C.1) $P_\gamma = 0$ if $P_\gamma < \lambda$;

(C.2) $\texttt{OMV}_{x,y} = \vec{0}$ if ($x = 0$ or $x = T_C - 1$) and $T_C > 2$;
$\texttt{OMV}_{x,y} = \vec{0}$ if ($y = 0$ or $y = T_R - 1$) and $T_R > 2$.

With regard to the time complexity to compute the OMV vector, the sums in Formula 5.2 are decisive and result in $\mathcal{O}(T_C \cdot T_R) = \mathcal{O}(n)$ with n being the total number of elements of the ROI mask matrix T, i.e., the calculation is linear to the number of ROI mask matrix elements. Indeed, an $\texttt{OMV}_{x,y}$ vector in Formula 5.1 is calculated in $\mathcal{O}(1)$ as all $MV^\gamma_{x,y}$ and all P_γ (with $\gamma = 1 \ldots 4$) are determined in $\mathcal{O}(1)$ for arbitrary $T(x, y)$. For $MV^\gamma_{x,y}$ this is proven by the time-complexity analysis in the previous section. The P_γ values are calculated by simple straightforward computations, hence $\mathcal{O}(1)$. In addition, these P_γ values are actually calculated only once as they are equal for all elements of the ROI mask.

Object resizing

Object motion captures the global motion of the object inside the ROI. In addition to this motion, it is also possible that the "size" of the object changes, due to for example camera zooming or a change in the relative distance of the object to the camera. As a result, the ROI must also change in size, otherwise the ROI will be too small or too large for the larger or smaller object respectively. We call this *ROI resizing*.

The pseudo-code listing to perform object resizing as explained in this section can be found in Listings B.6 and B.7.

In total there are two times three possible operations the size of a ROI undergoes: (1.a) reduction in width; (1.b) stay equal in width; (1.c) enlarge in width and (2.a) reduction in height; (2.b) stay equal in height; (2.c) enlarge in height. For every frame, one action is selected from (1.a), (1.b), and (1.c) and one action is selected from (2.a), (2.b), and (2.c). Both actions are executed as explained further in this section.

The first step to enable automated resizing of the ROI, is to determine the two appropriate actions. To do so, the change of the boundaries of the ROI is calculated using the following formulas. Note, $\texttt{OMV}^1_{x,y}$ represents the first vector component of the vector $\texttt{OMV}_{x,y}$, hence the horizontal motion; $\texttt{OMV}^2_{x,y}$ is the second vector component of this vector, hence the vertical motion:

$$\begin{aligned} M_L &= \sum_{y=0}^{T_R-1} \texttt{OMV}^1_{0,y}/T_R \\ M_R &= \sum_{y=0}^{T_R-1} \texttt{OMV}^1_{T_C-1,y}/T_R \\ M_H &= M_L - M_R \end{aligned} \tag{5.3}$$

$$\begin{aligned} M_U &= \sum_{x=0}^{T_C-1} \texttt{OMV}^2_{x,0}/T_C \\ M_D &= \sum_{x=0}^{T_C-1} \texttt{OMV}^2_{x,T_R-1}/T_C \\ M_V &= M_U - M_D \end{aligned} \tag{5.4}$$

M_L represents the horizontal motion of the left-hand ROI boundary, M_R is the horizontal motion of the right-hand ROI boundary, M_U is the vertical motion of the topmost boundary, and M_D denotes the vertical motion of the bottom boundary. M_H is the horizontal motion and M_V is the vertical motion.

Next, the two ROI resize actions are determined by the formulas:

$$\text{Action}_1 = \begin{cases} (1.\text{a}) & : & M_H > \delta \\ (1.\text{b}) & : & -\delta \leq M_H \leq \delta \\ (1.\text{c}) & : & M_H < -\delta \end{cases} \tag{5.5}$$

$$\text{Action}_2 = \begin{cases} (2.\text{a}) & : & M_V > \delta \\ (2.\text{b}) & : & -\delta \leq M_V \leq \delta \\ (2.\text{c}) & : & M_V < -\delta \end{cases} \tag{5.6}$$

δ is a threshold the motion of the ROI boundaries has to reach before the reducing or enlarging of the ROI mask takes place.

Knowing the required actions, the resizing of the ROI can take place. In case of action (1.a), the ROI matrix T is replaced by a new matrix T' with dimensions $((T_C - 1) \times T_R)$. The shifting values of the new matrix T' are a linear combination of the values of matrix T, expressed by the Formula 5.7.

$$T'(x, y) = \text{round}\left(\frac{(T_C - x - 1)T(x, y) + (x + 1)T(x + 1, y)}{T_C}\right) \tag{5.7}$$

where

$$x = 0, 1, \ldots, T_C - 2, \quad y = 0, 1, \ldots, T_R - 1.$$

This formula reduces the ROI matrix T by one column. Also, the location of the enhanced ROI mask is translated so the center of the ROI mask stays on the spot, hence $L_X = L_X + 4$. Next, we calculate the *residue* as $M'_H = M_H - 8$. If the residue value M'_H is still $> \delta$, the reduction procedure must be repeated. A similar matrix resize operation is executed in case of action (2.a) resulting in a new matrix with dimensions $(T_C \times (T_R - 1))$ and the new location $L_Y = L_Y + 4$. This is repeated as long as the residue value $M'_V > \delta$.

In case of operations (1.b) and (2.b), the ROI matrix T remains invariant.

Finally, in case of an enlarging size, the ROI matrix T is replaced by a new matrix T' with dimensions $((T_C + 1) \times T_R)$ and $L_X = L_X - 4$ in case of (1.c) and $(T_C \times (T_R + 1))$ and $L_Y = L_Y - 4$ in case of (2.c). The values of new matrix T' are a linear combination of the values of matrix T, analogue to the Formula 5.7. This enlarging must also be repeated as long as the residue values $M'_H < -\delta$ and $M'_V < -\delta$.

After applying the appropriate resizing actions, the residue values M'_H and M'_V are between $-\delta$ and δ. This value is added to the results of the calculation of Formula 5.3 and Formula 5.4 for the next frame, in other words we accumulate the motion of the ROI boundaries over multiple frames.

The time complexity of the algorithm to resize the ROI is determined as follows. First, the algorithm calculates the horizontal and vertical

motion (i.e., M_H and M_V) according to the Formula 5.3 and Formula 5.4. These formulas aggregate over respectively the number of rows and the number of columns of the ROI mask. This results in a time complexity of $\mathcal{O}(T_R)$ for Formula 5.3 and $\mathcal{O}(T_C)$ for Formula 5.4. The outcome of the formulas is used to determine the resizing action of the ROI according to the Formulas 5.5 and 5.6. With regard to the time complexity, this finding does not alter the time complexity.

The actual resizing of the ROI matrix depends on the type of action. If no resizing occurs (actions (1.b) and (2.b)), no additional steps must be performed, hence $\mathcal{O}(1)$.

In case of a vertical reduction (action (1.a)), the Formula 5.7 must be executed. This formula creates a new matrix in $\mathcal{O}((T_C - 1) \cdot T_R) = \mathcal{O}(T_C \cdot T_R) = \mathcal{O}(n)$, i.e., linear to the number of elements in the ROI matrix. For all other actions ((1.c), (2.a), and (2.c)), the time complexity can be deduced in a similar way, resulting in $\mathcal{O}(n)$ for each of these actions. When combining two resize actions or a resize action that must be repeated several times, the overall time complexity remains $\mathcal{O}(n)$.

Adding the time complexity $\mathcal{O}(T_R)$ to calculate Formula 5.3 and the $\mathcal{O}(T_C)$ to calculate Formula 5.4 to the time complexity for an actual resize action ($\mathcal{O}(n)$), the time complexity for the ROI resizing algorithm is $\mathcal{O}(n)$. If no resizing is done ($\mathcal{O}(1)$), the time complexity is $\mathcal{O}(T_R)$ or $\mathcal{O}(T_C)$, thus linear to the maximum number of rows or columns of the ROI mask matrix.

Cloaking

The algorithms discussed previously enable the creation of an ROI mask T, containing elements $T(x, y)$ that represent shifting values for blocks of 8×8 pixels. By setting a shift value to zero, it is possible to create arbitrary-shaped ROI masks. However to determine the object motion as described in previous section, the motion vectors for all $T(x, y)$ elements are determined and used in the overall object motion vector calculations. If the object inside the ROI is not rectangular, the calculation of the `OMV` uses all motion vectors within this ROI, even those that are not associated with this object. This is not desirable as these are associated with another object which might exhibit another motion trajectory. In addition, sometimes it is desirable to differentiate between the area to improve the visual quality and the object that is being tracked as the

creation of a region slightly larger than the object itself improves visual perception.

To solve these concerns, an additional *cloaking* layer is used that allows full separation of the visually important region and the object that is being tracked.

The cloaking layer is represented by a new matrix C. It has the identical dimensions as matrix T and consists of binary values indicating whether the $\texttt{OMV}_{x,y}$ vector must be taken into account in the overall object motion vector calculations. The determination of the values in the matrix C is done by a manual selection, similar to the determination of the object that must be followed, i.e., the initial determination of matrix T and the coordinates (L_x, L_y). This results in a third constraint for Formula 5.1:

(C.3) $\texttt{OMV}_{x,y} = \vec{0}$ if $C(x, y) = 0$.

Similar as for constraint (C.2), the divisor in Formula 5.2 is decreased by the number of matrix elements that fulfill constraint (C.3).

In Figure 5.14 a part of a frame is shown from the "hall monitor" sequence (see Figure 5.15 for the complete frame). The white grid represents the borders of the macroblocks, the black grid is the enhanced ROI mask matrix T and the cloaking matrix C, which coincide. The hatched parts indicate the cloaking matrix elements larger than zero. Only the motion vectors of the hatched parts will be used to determine the overall object motion $\texttt{OMV}$.

For the macroblock selection and object tracking algorithms, nothing changes when adding the cloaking matrix C, except for the overall object motion Formula 5.2, which receives the additional constraint (C.3) and the ROI resizing algorithm is extended in such a way that not only matrix T is adjusted to the new size, but also cloaking matrix C. This is done similar to the creation of matrix T' as discussed in the previous subsection resulting in a new matrix C'.

The cost in terms of time complexity for adding an additional cloaking layer only occurs during the resize operation. As the same resizing formulas are executed as for the ROI mask matrix T, the time complexity is the same. Hence, the time complexity of resizing the ROI matrix T and the cloaking matrix C is $\mathcal{O}(2n) = \mathcal{O}(n)$.

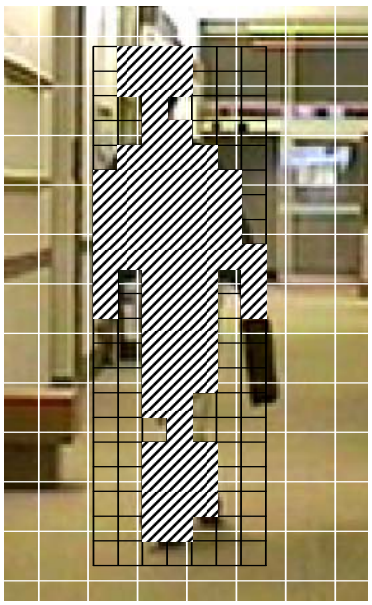


Figure 5.14: A part of a frame of the video sequence "hall monitor." The black grid represents the cloaking layer, which coincides with the enhanced ROI mask matrix T. The hatched parts indicate the cloaking matrix elements larger than zero. The motion vectors of the parts that are not hatched will not be used to determine the overall object motion `OMV`.

Analysis of the Time Complexity

To conclude this section, an overall time-complexity analysis of our algorithms is presented. The algorithms enabling automatic object tracking in the compressed domain are based on two principles: *object motion* and *object resizing* – resulting in *ROI resizing*.

For every frame, both algorithms are executed. As demonstrated in the previous sections, both have a (worst-case) time complexity of $\mathcal{O}(n)$, meaning that the time complexity is linear to the total number of elements n in the ROI mask matrix. Because both algorithms are sequential, the global time complexity for our algorithms is also $\mathcal{O}(n)$.

As such, using our lightweight object tracking algorithms implies adding an additional time-cost that is linear to the size of the object we want to track. Note, the size of the object is normally smaller than a complete frame. Furthermore, performance analysis of the MPEG-4 reference

software encoder and the MPEG-4 FGS reference software encoder in [132] and [133] shows that the encoder needs tens of millions operations per second. As such, our lightweight algorithms are only a very small fraction of the total required encoding time. Hence, object tracking in the compressed domain using the presented algorithms is feasible, even for real-time streaming applications.

Finally, it must be noted that the object tracking algorithms do not add any complexity to the video decoder. All algorithms, and associated time complexity, are part of the video encoder.

5.4.2 Evaluation, Results, and Discussion

Methodology

In this section we evaluate the accuracy of our lightweight object tracking algorithms. First, we created a video of a playing card moving from right to left against a non-uniform background, while the camera is zooming in. The first frame shows the playing card at the right hand side. Throughout the frames, the card moves towards and outside the left boundary of the frame, until only a small piece is visible in the last frame. Meanwhile the card has more than doubled in size due to the camera zooming operation. The video has a resolution of 320×240 resulting in 20×15 macroblocks and has a length of 501 frames. A picture from this video sequence is shown in Figure 5.10. We also used two well-know test sequences, namely "hall monitor" (with a CIF resolution of 352×288) and "crew" (with a high definition resolution of 1280×720). One can see excerpts of six frames of the former in Figure 5.15 and a still image of the latter in Figure 5.16. The first sequence was chosen because the object being tracked moves away from the camera (hence, the object is "scaled down"). The second sequence was chosen because of the larger resolution and higher complexity of the scene.

To test the algorithms, we need to compare our results to a "correct" set. We asked four different persons who have no visual defects to indicate for each frame the smallest possible rectangular region containing the object, i.e. the card for the first test sequence; the man on the left in "hall monitor" sequence and the man on the front right in the "crew" sequence. All macroblocks that hold (a part of) this region are marked and stored as the *object indication* for the given frame. The object indication for the first frame was given as an input, the remaining frames

Figure 5.15: Excerpts of six frames (no. 5, 15, 25, 35, 45, and 55) from the "hall monitor" test sequence.

Figure 5.16: A still image from the "crew" test sequence.

were tagged manually. From the four resulting sets of object indications, we distilled one reference set for each test sequence. In the next subsection, the construction of these sets is discussed.

The same video feed is used as input video sequence for the object tracking algorithm. It receives a matrix T with the initial location (L_X, L_Y) so the selection of macroblocks, as explained in Section 5.4.1, results in an object indication identical to the object indication that was given for the first frame for the manual marking. In the first sequence, the playing card fits perfectly in the ROI mask, hence the cloaking matrix C is completely filled with values larger than 0. For the other two sequences, the cloaking matrix C is used so that only the motion vectors of the human form are taken into account. Figure 5.14 depicts the cloaking matrix for the "hall monitor" sequence. All values of the matrix T are $\alpha = 9$. Next, different parameters were used to investigate their optimal settings and their influence on the results:

- *(C.1)*: $\lambda = 0.00$, $\lambda = 0.05$, $\lambda = 0.10$, $\lambda = 0.15$, $\lambda = 0.20$, $\lambda = 0.25$.
- *δ in Formula 5.5 and Formula 5.6*: $\delta = 2$, $\delta = 4$, $\delta = 8$, and $\delta = 16$.
- *applying and not applying (C.2).*

It is necessary that $\lambda \leq 0.25$ in constraint (C.1). If $\lambda > 0.25$ and the ROI mask is located in such a way that none of the overlapping percentages is larger than λ, then constraint (C.1) is never met and consequently `OMV` will be the null vector. As a result, the values L_X and L_Y do not change, and the ROI mask will not move. For all subsequent frames, the same event will occur, keeping the ROI mask at the same location.

All different combinations are encoded with an I(P*) GOP[5] structure. During the encoding, the macroblocks that are marked for selective enhancement by the algorithms are logged for each frame.

Results

First, we elaborate on the results of the manual object indication for the given video by comparing the different indications given by the four

[5] In this notation I(P*) means the encoded video starts with an I-frame, followed by all P-frames.

persons. To create the reference set for the first test sequence, 473 object indications can be selected in a trivial manner as there is a majority. To have for each of the 500 frames one object indication, 27 object indications must be added to the reference set. To do this, we determine the person that contributed the most indications to the 473 object indications and we select the remaining 27 object indications from this person. This results in a reference set of 500 object indications, for each frame one indication.

The test reference sets for the "crew" and "hall monitor" sequences are determined in a similar way.

It can be observed that selecting the "correct" region is, even for humans, a difficult task. Further investigation of the results shows that most often the four opinions only differ in one row and/or one column of macroblocks.

Next, the logs that were created during the encoding of the video are compared to the constructed reference set and compared to the reference set accepting a minor dissimilarity of one row and/or one column. The criterion for evaluation is the percentage of identical object indications: the higher this value, the better the automatic algorithm performs. This results in two measurements for each of the different test sequences:

(M.1) Compare to the reference set.

(M.2) Compare to the reference set and accept a dissimilarity of one row and/or one column as correct.

The results are shown in Figure 5.17 and Table 5.1.

Discussion

The relationship of the results for (M.1) and (M.2) is as expected. Obviously, the results that accept a dissimilarity of one row and/or one column as correct (M.2) are always better than the cases were such dissimilarity is not accepted (M.1).

We observe that the value of λ does not have a significant influence on the results. Because Formula 5.1 weights the value of the motion vector $MV^{\gamma}_{x,y}$ with overlapping percentage P_{γ}, the influence of small overlapping areas is automatically reduced.

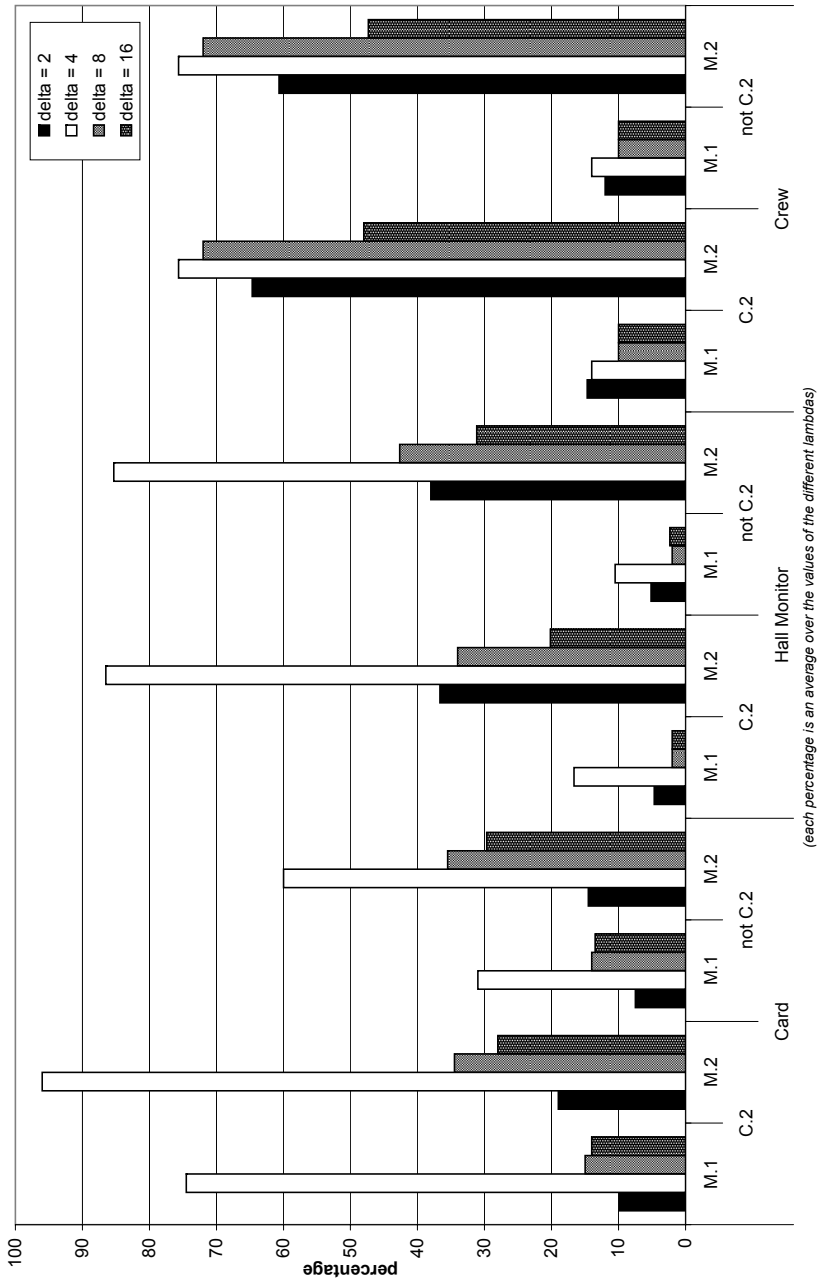


Figure 5.17: Object tracking test results.

Table 5.1: Object tracking test results (in %).

		Card				Hall Monitor				Crew			
		C.2		not C.2		C.2		not C.2		C.2		not C.2	
δ	λ	M.1	M.2	M.1	M.2	M.1	M.2	M.1	M.2	M.1	M.2	M.1	M.2
2	0.00	9	18	8	13	5	36	5	37	18	68	14	70
	0.05	8	18	8	13	5	36	5	36	16	68	14	66
	0.10	10	19	8	14	5	36	5	36	16	68	14	68
	0.15	10	19	7	14	5	36	5	37	16	70	8	48
	0.20	11	20	7	14	4	38	5	38	8	46	8	42
	0.25	11	20	7	19	4	38	6	44	14	68	14	70
4	0.00	75	96	30	59	11	85	10	85	14	78	14	78
	0.05	75	96	30	60	16	87	10	85	14	78	14	78
	0.10	76	96	31	61	12	86	10	85	14	78	14	78
	0.15	76	96	32	60	21	88	10	85	14	76	14	76
	0.20	74	96	30	60	20	86	11	86	14	72	14	72
	0.25	71	96	33	60	20	87	12	86	14	72	14	72

Table 5.1: Object tracking test results (in %) (continued).

		Card				Hall Monitor				Crew			
		C.2		not C.2		C.2		not C.2		C.2		not C.2	
δ	λ	M.1	M.2	M.1	M.2	M.1	M.2	M.1	M.2	M.1	M.2	M.1	M.2
8	0.00	15	34	14	36	2	34	2	42	10	72	10	72
	0.05	15	34	14	36	2	34	2	42	10	72	10	72
	0.10	15	34	14	36	2	34	2	42	10	72	10	72
	0.15	15	36	14	36	2	34	2	44	10	72	10	72
	0.20	15	35	14	35	2	34	2	42	10	72	10	72
	0.25	15	34	14	34	2	34	2	44	10	72	10	72
16	0.00	14	28	14	29	2	20	2	26	10	48	10	48
	0.05	14	28	14	29	2	20	2	26	10	48	10	48
	0.10	14	28	14	30	2	20	2	25	10	48	10	48
	0.15	14	28	13	30	2	20	4	60	10	48	10	48
	0.20	14	28	13	30	2	21	2	25	10	46	10	46
	0.25	14	28	13	30	2	20	2	25	10	46	10	46

We also observe that our algorithms are also insensitive to sudden motion changes. Indeed, if the object stops, the calculation of the Object Motion Vector will result in a null vector as the motion vectors associated with the object will also be the null vector. Hence, the ROI mask will not be displaced. If the object starts moving again, the motion vectors will reflect this motion and the Object Motion Vector will reflect the displacement of the object, regardless of the direction of this last motion. In fact, the object that is being tracked in the "hall monitor" sequence stops for a few frames and continues to move in another direction.

The table also shows the influence of parameter δ. The parameter δ determines how fast the matrix T must be resized. The necessity of this parameter is proven by the fact that the results for $\delta = 4$ are superior. This parameter is independent of the kind of resizing, namely an object enlarging in the "card" and "crew" sequences and an object reduction in the "hall monitor" sequence. The biggest difference is shown when (C.2) is enabled. For the "card" sequence, the improvement for using $\delta = 4$ instead of $\delta = 8$ ranges from 56.6% to 61.8%. When comparing $\delta = 4$ to $\delta = 16$ the difference is even more clear. Setting the parameter $\delta = 8$ or $\delta = 16$ makes the resizing algorithm too slow to react properly. In case of the "card" sequence, this usually means that the matrix T is too small as the algorithm does not counter the zoom in operation immediately. δ is set to the optimal value of $\delta/2 = 4$. In case of the "crew" sequence, the *enlarging* of the traced object itself occurs more steadily and slowly, when compared to the playing card of the first test sequence. Hence the need to resize in two consecutive frames is rare; the resizing is more distributed over multiple frames. As such, if the algorithm misses a resize for one frame, it can correct this in the next frame. This explains the smaller influence of the parameter δ. Setting δ too small (e.g., $\delta = 2$) results in an unsteady resizing.

The table also reveals the big impact of constraint (C.2) for the "card" sequence, in particular when δ is set to 4. It is clear from the test that the border of a ROI mask is less reliable, especially after some iterations. Constraint (C.2) is never applied if the ROI mask is too small in size, hence the conditions $T_C > 2$ and $T_R > 2$, otherwise it is possible that the `OMV` is always the null vector. Constraint (C.2) must always be enabled in combination with $\delta = 4$ as this generates better results.

However the latter depends on the content of the video itself. As the test video has a growing object due to a camera zooming operation, setting δ larger than the optimal value, will result in a too small matrix

T. Therefore, if the matrix T is smaller than the real object, adding the motion vectors of the elements on the edges of matrix T, means incorporating the motion vectors of the actual object being tracked; this is good. For video sequences where the matrix T is larger than the actual object, not enabling constraint (C.2) gives a worse result as the motion vectors outside the actual object are taken into account.

The reason we do not observe this large influence for the two other sequences is because of the usage of the cloaking matrix C. Indeed, because the used cloaking matrix – such as the one depicted in Figure 5.14 – removes most macroblocks that are part of the border, the enabling or disabling of constraint (C.2) hardly influences the results for these sequences.

To conclude, the optimal settings for a video sequence are:

- *(C.1)*: $\lambda \leq 0.25$
- *(C.2)*: enabled or use cloaking matrix C
- δ: 4

Our automatic algorithm reaches near perfection when comparing to the reference set and allowing one row or column mismatch (M.2) for the "card" sequence: 96.1% on average over the six possible values for λ. Comparing to the reference set without allowing one row or column mismatch (M.1) gives an average result of 74.5% for this first test sequence.

For the "hall monitor" sequence, using the optimal settings our algorithms achieve on average 86.5% when allowing one row or column mismatch (M.2). However, if a mismatch is not allowed (M.1), the results are much worse in comparison to the first sequence. The good results for (M.2) indicate that mostly there is only one row and/or one column mismatch. Looking more into detail we see that the automatic algorithm marks the object too large, hence one column or one row too much. While this restrains the percentages for (M.1), the visual perception is not negatively influenced by enhancing a little more than the object. Furthermore, the mismatch does not propagate over successive frames so the algorithms are still useful.

Finally, the "crew" sequence achieves also a good result when using the optimal settings: on average 75.7% when allowing one row or column mismatch (M.2). While these results are lower than the "card"

sequence – particularly when comparing to (M.1) – it must be noted that this sequence is far more complex as it contains more (moving) objects, a moving background with similar texture and colorization of the object that is being tracked, abrupt luminance changes due to flash photography and so on.

5.5 Related Work

In Section 5.3, we discussed the most recent scalable video coders. An older scalable video coding technology is MPEG-2. It is actually the first widespread video coder that natively supports SNR and spatial scalability (and temporal scalability like most video coders). It uses a two-layer approach: a base layer contains the coarsest or smallest version and an enhancement layer improves the quality or resolution thereof. MPEG-2 supports a scheme that combines both types of scalability, resulting in a three-layer system. However, as the scalability features of this coder are infrequently exploited and surpassed by more advanced technologies, we did not discuss this coder in detail. For more information about MPEG-2 and its scalability features, we refer the reader to [75, 134].

In Section 5.4, we discussed a fast object tracking scheme. In the literature many algorithms, systems, and techniques have been described to identify and track objects in video streams. We first discuss object segmentation, followed by object tracking algorithms.

Most discussed techniques for object segmentation use object models and ontologies [135], color information (e.g., flesh tone for human face recognition) [114, 136], semantic and probabilistic decomposition of the video frames with learning capabilities [137, 138], temporal comparisons between consecutive images of a video stream [139], leveled watershed techniques [140], or a combination of the previous techniques as in [141, 142]. These techniques could be used independently to track an object over the consecutive frames of a video stream, but this implies a computationally burden.

The object tracking techniques can be subdivided into object tracking in the pixel domain and object tracking in the compressed domain. The former technique requires that the video stream is decoded before it can be processed. Tracking in the pixel domain allows to apply advanced techniques like stochastic algorithms [143], but the necessary decoding step decelerates the tracking process and makes it infeasible for real-time

applications.

The techniques for object tracking in the compressed domain take advantage of the motion vector field that the video encoders calculate during the encoding process of the sequence. In [144], trajectory estimation is made based on the motion vector field and partially decoded DCT coefficients. While this technique promises to be "fast enough for real-time applications," it assumes that there is "no camera motion." A similar technique is discussed in [145], but fails when "the object was small or the object was non-rigid or was changing a lot in shape and size." In [146], the camera motion is estimated using a Hough transform for the overall motion and a mean-shift algorithm based on the motion vector field. The technique has good results for shorter sequences, unfortunately the computations are rather complex and time-consuming, making this technique difficult to implement for real-time applications. Finally, we want to mention the work of Favalli et. al in [147]. In this work it is described how object tracking can be done, solely using the information of the motion vectors and applying very simple calculations, hence adding "as little additional processing as possible to the complexity of a standard decoder."

Our object tracking scheme presented in this chapter is different from the above discussed techniques, and from [147] in particular as it adds no complexity to the decoder, and only very little additional computations with low time complexity to the encoder of the video stream. By doing so, our techniques can be used for real-time streaming applications in contrast to [146]. Also, the discussed techniques allow the tracking of any kind of object of any arbitrary shape and are capable of handling camera motion, object resizing, and object deformation, addressing the main shortcomings of [144] and [145]. Furthermore, our techniques are independent of the video encoding specification; for demonstration and testing purpose we used the MPEG-4 FGS codec and exploited its Region-of-Interest capabilities to visualize the results of the object tracking algorithms. Finally, the novel methods for object tracking in the compressed domain are detached from the macroblocks by introducing two independent layers on top of the macroblock grid. This enables the tracking of fine-detailed objects.

5.6 Conclusions and Original Contributions

In this chapter, we have discussed the various forms of scalability for video streams, namely temporal scalability, SNR scalability, and spatial scalability. We gave an overview of the different video coders that have native support for one or more types of scalability. Special attention was paid to the Region-of-Interest coding and we illustrated how ROI can be accomplished in the MPEG-4 FGS codec.

Finally, we have discussed novel algorithms allowing a video coder to automatically track objects and implemented this functionality in MPEG-4 FGS to enhance its ROI capability. Our object tracking algorithms have a very low time complexity that is linear to the size of the object, making them very useful for real-time streaming applications. We use the motion vector field calculated by the encoder's motion estimation algorithms to capture the overall motion of the tracked object. We also introduced an algorithm that allows an encoder to cope with the "enlargement" and "shrinking" of an object. All our algorithms are capable of tracking any kind of object in a video stream. Furthermore, our algorithms are not bound to the (relatively large) size of a macroblock; we use a fine grid on top of a frame so the algorithms can track an object in a more detailed way.

All our novel methods presented here are generic and can be implemented in any codec that estimates the motion vector field. For testing and evaluation purposes, we have implemented the algorithms within the MPEG-4 FGS reference software encoder. We used the ROI capabilities of this codec to visualize the results of the algorithms. A second layer was introduced enabling us to differentiate between the Region-of-Interest and the (possibly smaller) arbitrary-shaped objects within this region. Three test sequences were used to evaluate the influence of various parameters. The results of our algorithms were compared to manually constructed reference sets so an optimal parameter set was determined.

From these results, we can conclude that these novel lightweight algorithms are capable of tracking objects in complex scenes; can handle scaled down or scaled up objects; and are independent of the resolution of the video stream.

Our algorithms have a very low time complexity, hence, they do not prevent the video encoder to work in real-time. However, the MPEG-4

FGS reference software that we used in our experiments is too slow for real-time encoding as well as for real-time decoding. We did not succeed in creating a real-time MPEG-4 FGS-compliant encoder or decoder. We also did not succeed in creating a system that facilitates the identification of the object that must be tracked nor incorporating such an existing system.

The research that has resulted in this chapter of this thesis is also discussed in our following publications.

1. Koen De Wolf, Robbie De Sutter, Wesley De Neve, and Rik Van de Walle. Comparison of Prediction Schemes with Motion Information Reuse for Low Complexity Spatial Scalability. In *Proceedings of SPIE/Visual Communications and Image Processing*, volume 5960, pages 1911–1920, Beijing, China, July 2005

2. Robbie De Sutter, Koen De Wolf, Sam Lerouge, and Rik Van de Walle. Lightweight Object Tracking in Compressed Video Streams Demonstrated in Region-of-Interest Coding. *Eurasip Journal on Applied Signal Processing*. To appear

3. Davy De Schrijver, Wesley De Neve, Koen De Wolf, Robbie De Sutter, and Rik Van de Walle. An Optimized MPEG-21 BSDL Framework for the Adaptation of Scalable Bitstreams. *Journal of Visual Communication and Image Representation*. To appear

Chapter 6

Integration and Concluding Remarks

6.1 Integration

Before concluding this thesis, we first discuss how to integrate the previously discussed techniques in order to create a UMA-compliant Video-on-Demand application with support for time-varying metadata. The goal of this application is to stream audio-visual content selected by an end user over an IP-based network such that the content is dynamically and on-the-fly optimized and adapted to the consumption context.

We briefly discuss the techniques that we have used in order to create this VoD application hereafter. A detailed report and discussion on its creation can be found in Appendix C. A demonstration version of this application is available at the Multimedia Lab research group Website[1].

To start, a client application invokes a web service at a broker service in order to retrieve an overview of the available audio-visual streams and this by using the SOAP technology discussed in Chapter 3. The end user selects a stream through a *graphical user interface* (GUI) and his selection invokes another web service at the broker service. Simultaneously, the context information is collected and structured using the MPEG-21 DIA-UED software toolkit which results in an MPEG-21 DIA-UED compliant XML-data stream (Chapter 2). In order to reduce the overhead,

[1]The Multimedia Lab research group Website is available at `http://multimedialab.elis.ugent.be`.

the XML stream is compressed using the BiM technology, which was discussed in Chapter 4. The broker processes the context information using our serialization-agnostic parser, also discussed in Chapter 4, and determines the adaptation rules. The client retrieves the location of the streaming server and initiates the streaming request. The streaming server processes the adaptation rules, adapts the audio-visual content using the scalability techniques discussed in Chapter 5, and starts streaming the optimized content. Furthermore, the streaming server is capable of handling updates of the adaptation rules – hence, indirectly also updates of the context information – dynamically and on-the-fly in order to realize the support for time-varying metadata, discussed in Chapter 3. As such, the end user receives content optimized to the (changing) context environment. More detailed information on the architecture, usage scenario, technologies, and implementation of the application can be found in Appendix C.

The creation of this application illustrates the usability of the previously discussed techniques. Nevertheless, some issues were revealed. The main issue was the lack of real-time decoders for scalable encoded video content; we selected AVC-encoded content as an alternative. As a consequence, our on-the-fly content adaptation is limited to temporal scalability and enabling/disabling of the audio stream. The more advanced scalability schemes – such as the selective degradation of the video using Region-of-Interest enhanced with the fast object tracking technique – could not be demonstrated in this application. Using MPEG-B BiM as the alternative serialization method resulted in a second issue. To BiM encode XML-based data, we use the Windows-based BiM reference software. However, as the network is emulated by software running on a Linux-based operating system, the reference software could not be used. Unfortunately, there are currently no (commercial or open source) alternative BiM encoders available. Final and third issue, for demonstration purposes we allowed the end user to manually alter the context information. Although this is acceptable and desirable to illustrate the feasibility of the dynamic and on-the-fly adaptations, real-life applications should aggregate the context information automatically. However, no generic solutions for realizing this currently exist.

Notwithstanding the aforementioned issues, we are convinced that this Video-on-Demand application illustrates the usability of our results and our novel contributions discussed in the previous chapters.

6.2 Concluding Remarks

As the Internet is continuously expanding with new content that is consumable on new devices and that can be transmitted over new types of networks, actions are needed to make transparent and ubiquitous content consumption possible anywhere, anytime, and anyhow. Currently ad hoc solutions try to make this happen, however these solutions are inadequate and unsustainable in the long run. Take, for example, an Internet-based *Video-on-Demand* application, hence, the streaming of audio-visual content over IP-based networks. This kind of multimedia content consumption is becoming more and more popular as illustrated by rapid growing number of *video blogs*. Currently, if an end user wants to consume audio-visual content over the Internet, he first has to answer several technical questions about his device, the network connection, and his personal preferences – in short, questions about the consumption *context*. As a result, he can select a particular version of the video that more or less suits the context. The content provider has a *simulstore* containing a limited number of semantically equal videos with different technical characteristics, for example audio-visual content streams with different resolutions and frame rates. Hence, the end user receives suboptimal content while the content provider has to maintain several similar content streams.

As more and more different kinds of end-user devices and network technologies become available to acquire and consume audio-visual streams, this solution is no longer acceptable. Indeed, either the content provider must support an increasing number of different kinds of versions of the same content or the consumer must be satisfied with content that is less suited for his particular context.

The *Universal Multimedia Access* framework provides an answer for these problems. The core idea is to enable the consumption of multimedia content for different usage contexts (hence, different end-user interests and profiles, end-user devices, networks, and content providers) by creating different representations of the same information from a single content base. In other words, it adheres to the "create once, play everywhere" paradigm.

In this thesis, I have investigated the requirements to realize this UMA vision by investigating the different parts required to create a UMA-compliant VoD architecture.

In Chapter 2, I studied how to describe in a standardized way *what* is being consumed – i.e., the *content* – and *how* this is being consumed – i.e., the *context*. After studying the work methods used by the Flemish television broadcasters in order to describe their content, it became clear that an evaluation framework was needed to compare the international standards for content description. I created such a framework consisting of four main criteria that can be used to determine the most optimal content-description standard for a particular usage. My evaluation framework is generic and can be used to compare any content-description technology. This is demonstrated by applying the evaluation criteria to DCMES, P/Meta, MPEG-7, and SMEF – these are international standards to describe audio-visual content. The evaluation framework demonstrated that MPEG-7 is the most appropriate standard to annotate audio-visual content if this content information will be used in UMA-compliant architectures. Second, I gave an overview of the currently available context-description standards, namely HTTP Headers, CC/PP, and MPEG-21 DIA-UED. My study indicated that the latter is the most generic and comprehensive standard. Although this standard is vast, I created software that runs on very constrained devices (such as cell phones) that is able to read, modify, and write UED-compliant messages. As such, thanks to my software, application developers can easily and rapidly develop applications that are MPEG-21 DIA-UED compliant without the need to learn the specific UED syntax and structure. My software toolkit was submitted and accepted by the MPEG consortium as the reference software tool for the MPEG-21 Digital Item Adaptation – Usage Environment Description tool.

The information about the content and the context is used by a *content adaptation system* in order to make adaptation of the content possible such that the resulting optimized version is consumable in the given context. In Chapter 3, I decided to tackle this by dividing the adaptation process into a *content adaptation decision engine* and a *content adaptation engine*. The former exploits the information about the content and the context to determine how to adapt to content, i.e., the *adaptation rules*; the latter performs the actual adaptation of the content. Next, I decided on the location of these two engines in a UMA-compliant architecture: the content adaptation decision engine is placed as a separate entity and the content adaptation engine is located near the content at the content provider. Also, I investigated how to *negotiate* the content and context information with the content adaptation decision engine. By making abstraction of this decision taking process and regarding

it as a web service. To invoke these web services, I investigated the two most suitable techniques, namely XML-RPC and SOAP. Finally, I discussed my important extension to the UMA framework, namely the *time-varying metadata* concept. The main idea is to dynamically optimize the content to a *changing* context by re-negotiating the context information. As such, the modifications to the consumption environment are handled automatically and result in the re-optimization of the content on-the-fly.

Chapter 4 solves an important issue that came to light in the two previous chapters, namely due to the usage of XML to describe the content and context information, overhead is introduced with regard to bandwidth usage. Indeed, not only is XML-based data marshalled as verbose plain text, XML also does not have any provisions to update information as needed to support the time-varying metadata concept. In this chapter, I first performed a study of the different kinds of *XML parser models*. This research allowed me to decide upon the best model to support XML updates, and hence time-varying metadata. As a result, I created a *serialization-agnostic XML parser* according to the *Cursor Model*. My parser is able to handle traditional plain-text XML data as well as non-textual encoded XML data. As such, applications can use my parser to handle XML data without being aware of the actual encoding format. Hence, it shields the users from the technical details of the encoding format so the usage of non-textual encoded XML data is transparent and does not increase complexity for the user. Next, I studied potential *alternative (non-textual) XML serialization formats*, namely ZIP compression, ASN.1-PER, and MPEG-B BiM – the latter supports XML updates natively. These techniques were evaluated to their applicability to address the overhead concerns when negotiating the UED-based context information and for the XML and Internet-based *RSS* application. Results show that the BiM technology is most efficient in terms of overhead reduction (i.e., *compression ratio*), but worst in terms of executing time performance. The latter is due to the usage of non-optimized reference software. Unfortunately, I was unsuccessful in creating an optimized BiM encoder and decoder myself. ZIP compression is ranked as the second best option because its compression ratio is only a little bit lower in comparison to BiM. Furthermore, the ZIP compressed data must be completely received before decompressing and parsing can occur, while BiM data can be parsed immediately in the compressed domain even if the data is not yet completely received. On the other hand, the ZIP compression technique is very widespread and

introduces a very low delay during processing. Hence, I believe the BiM technology is very promising but only practically usable if optimized encoders and decoders become available. Until then, ZIP compression is a more than acceptable alternative.

The discussed technologies in the first four chapters make it possible to provide a content adaptation decision engine with content and context information in a standardized and optimized way. Using this information, the engine decides upon the adaptation rules. According to the UMA principles, the optimized content should be derived from a single content base. Chapter 5 starts with an overview of *video scalability* techniques that makes it possible to manipulate audio-visual streams in the compressed domain so a new stream can be obtained with other (technical) characteristics. The three main types of scalability were discussed, namely *temporal scalability*, *signal-to-noise ratio scalability*, and *spatial scalability*. Next, an overview was given of the most recent video coding technologies that natively support the signal-to-noise ratio scalability and/or the spatial scalability techniques, namely MPEG-4 FGS, MPEG-4 SVC, and Wavelet-based video coding. Special attention was paid to the *Region-of-Interest* technique in MPEG-4 FGS, which is a particular form of SNR scalability. A ROI is an area within the video that is seen as more important than the remaining area. As such, if a quality reduction of the video is required (for example, to comply with a given bit rate), the area outside the ROI degrades before the area inside the ROI. In the remainder of Chapter 5, I discussed my novel technique that allows fast object tracking within the compressed domain by reusing the motion vector field. My algorithms not only allow the tracking of an object, but are also capable of handling the resizing thereof. To illustrate the applicability of the algorithms, I have implemented them into the MPEG-4 FGS encoder. Several tests were performed to demonstrate the quality of the tracking algorithms. I also proved that my algorithms are fast enough to work in real-time by analyzing thoroughly the time complexity of the algorithms.

Finally, I briefly discussed the integration of the techniques described in the previous chapters by creating a Video-on-Demand application that is compliant with the UMA principles – i.e., streaming of audio-visual content optimized to the consumption context – and that can handle time-varying metadata – i.e., on-the-fly re-optimizing of the content to a changing context. A detailed report on the construction of the VoD application can be found in Appendix C.The lack of a real-time advanced

scalable video decoder (for example, an MPEG-4 FGS decoder) was revealed as a major problem to realize the VoD application. As such, I was obliged to use a more traditional video codec, in particular the H.264/MPEG-4 AVC codec. Hence, the application only optimizes video streams using temporal scalability.

There are some interesting research topics that are not discussed in this thesis, but that would be useful to investigate as an addition to the work I have presented. A first topic is the way to decide upon the adaptation rules. In my constructed application, I used a straightforward mapping algorithm, however in practice this is a multi-criteria optimization problem. On top of that, different end users might disagree on the meaning of "optimized content": one might prefer a video with a smaller resolution at full frame rate, while another might prefer a normal resolution at half frame rate. If both optimized versions are feasible for the given context, the determination of the adaptation rules is not as straightforward as a mapping algorithm.

Also the aggregation of the context information is a related research topic. The MPEG-21 DIA-UED tool and my software toolkit make it possible to easily express the context, however no generic solutions currently exist to automatically retrieve this information from the network, the end user, and his device.

A third topic is about the time-varying metadata. In this thesis, I assumed that each time an update of the time-varying metadata occurs, this is signaled to the content adaptation decision engine. However, if the context is very fluctuating, for example unsteady network bandwidth, sending this information each time would result in a very unstable adaptation of the content. Hence, a minimum threshold should be defined before the modified context information is transmitted. Research on the optimal threshold levels must still be performed.

A fourth research topic is related to the alternative XML serializations. Although these techniques solve the overhead concerns of XML, it also makes some XML tools unusable, most notably the XML transformation tools. Transforming alternatively serialized XML-based data in the binary domain is a challenging new research topic.

And finally, although research with regard to video scalability is very advanced, like my object tracking techniques, these techniques are unfortunately only suitable for research purposes. Real-time scalable video encoders and decoders are needed to convince content providers of the

advantages of these advanced scalability techniques.

Notwithstanding the aforementioned open issues, I hope to have convinced the reader that the techniques introduced in this thesis are useful to create and to ameliorate (Internet-based) multimedia applications and help to simplify the construction of real UMA-compliant applications.

The research that has lead to this thesis resulted in a number of publications. Two papers are accepted for publication in journals that appear in the Science Citation Index, namely in Springer's *Multimedia Systems* [2] and in Springer's *Lecture Notes in Computer Science* [3]. Two papers are currently under review for publication in journals that appear in the Science Citation Index, namely Eurasip's *Journal on Applied Signal Processing* [4], and Elsevier's *Journal of Visual Communication and Image Representation* [5]. In addition, I have contributed 11 papers to international conferences as first author [6–16]. Collaboration with fellow researchers resulted in 14 publications as co-author [17–30]. Finally, 10 contributions were submitted to the MPEG community [31–40].

Appendix A

MPEG-21 DIA-UED

In this appendix, we give background information on and examples of the MPEG-21 DIA-UED context description tool as introduced in Chapter 2.

First, we give a detailed overview of the different parts and subparts of the MPEG-21 DIA-UED tool.

Next, the UML class models are shown in Figure A.1 to Figure A.16.

Hereafter in Section A.3, several examples of usage context descriptions compliant to the UED specification are given. The first example is used for the tests to evaluate the UED software toolkit of Section 2.3.2. All examples are used for the evaluation of the alternative serialization formats in Chapter 4. Listing A.1 is the initial input for the Use Case 1; Listing A.2, A.3, and A.4 show the three updates.

Finally, this appendix shows an example on the usage of the API of the UED software toolkit, which is also used during the software toolkit test in Section 2.3.2.

A.1 Overview specification

A.1.1 User Characteristics

The first and largest part is the *User Characteristics* part. It contains information about the user, his preferences, and his intentions. Table A.1 gives an overview of the different subparts and their meaning.

Table A.1: UED – User Characteristics.

User Info:
stores generic information about the user, such as his name and contact address.

Usage Preferences:
defines the preference of the user on the usage of different types of content. For example, news content is preferred over entertainment.

Usage History:
gives information about the previous actions the user has undertaken for specific types of content. For example, record a news broadcast.

Audio Presentation Preferences:
stores the user preferences in regard to the audio component of a multimedia presentation, such as preferred volume, equalizer settings, and default output device.

Display Presentation Preferences:
contains information in regard to the visualization of a multimedia presentation. Examples are the preferences of the brightness, saturation, contrast, and color temperature.

Graphic Presentation Preferences:
determines the preferred modification in graphical quality by the user if such change is required. The user can declare his preference in regard to the geometry, texture, and animation.

Conversion Preferences:
if content is not suitable for a specific terminal, adaptation of that content may occur. By adding information in this subpart, the user can give information about its preferred way of converting content, for example, from video to images or to audio.

Presentation Priority Preferences:
if there are multiple methods to convert media, this subpart allows the user to express an order in these conversions that may also be dependent on the type of content. For example, sport content may be converted to audio, while news content must be converted to images.

Table A.1: UED – User Characteristics (continued).

Focus of Attention: stores information about the region the user is particularly interested in, for example, a news presenter instead of the news studio setting. This can be used to improve the quality for this region or to increase the content with additional information about the region.
Visual Impairment: contains information about the visual deficiency of the user, such as his form of blindness.
Auditory Impairment: contains information about the hearing deficiency of the user, such as hearing loss in the left and/or right ear.
Mobility Characteristics: stores information on the movement of the user, for example, his direction.
Destination: stores the final destination of the user. This can be used to enable context-aware services.

A.1.2 Terminal Capabilities

The second part of the UED tool is the *Terminal Capabilities* part. It contains information on the hardware specifications of the end-user device. Its subparts are listed in Table A.2.

Table A.2: UED – Terminal Capabilities.

Codec Capabilities: specifies all encoding and decoding capabilities of the end-user device.
Display: contains information on the display(s) of the end-user device, for example, resolution, color capabilities, and refresh rate.

Table A.2: UED – Terminal Capabilities (continued).

Audio Output Capabilities: gives the number, type, characteristics, power, and other useful information on the speaker(s) of the end-user device.
User Interaction Possibilities: enlists the possibilities and characteristics of the input possibilities of the end-user device, such as information on the keyboard, mouse, and microphone.
Device Class: selects the type of end-user device from a list of possibilities.
Power Characteristics: gives information on the battery status of the end-user device, such as power and estimated remaining time.
Storage Possibilities: contains information on the storage capabilities of the end-user device, for example, available free space and transfer rate.
Data Input/Output Characteristics: gives information on all I/O-devices, such as the number of devices and bus speed.
Benchmark Information: stores information about the overall processing capacity of the end-user device by a benchmark rating.
IPMP Tools: gives an overview of the supported *Intellectual Property Management and Protection* (IPMP) capabilities of the end-user device.

A.1.3 Network Characteristics

Next, the *Network Characteristics* part describes the static network capabilities and the time-varying network conditions, hence it is composed of two subparts as shown in Table A.3.

Table A.3: UED – Network Characteristics.

Network Capability: stores static information about the theoretical characteristics of the network, namely minimal guaranteed available bandwidth, maximum available bandwidth, error rate, error correction capabilities, and guarantee for in-sequence delivery of packets.
Network Conditions: contains dynamic information about the actual status of the network, in particular feedback on the delay, error rate, and available bandwidth.

A.1.4 Natural Environments Characteristics

Finally, the UED tool is completed with the *Natural Environments Characteristics* part that describes the surroundings of the end user. Table A.4 gives an overview of the subparts of this final part.

Table A.4: UED – Natural Environment Characteristics.

Location: stores the actual location of the end user, either by his precise geographical location (using longitude and latitude coordinates) or by a description (e.g., an address).
Time: tells the local time at the location of the end user.
Audio Environment: gives information on the noise at the location of the end user.
Illumination: stores information on the light intensity that hits the display of the end-user device.

A.2 Class Model

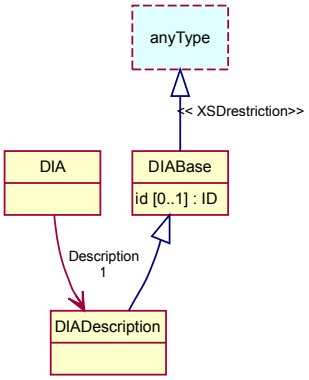


Figure A.1: MPEG-21 DIA-UED Class Model – DIA Base Data Type.

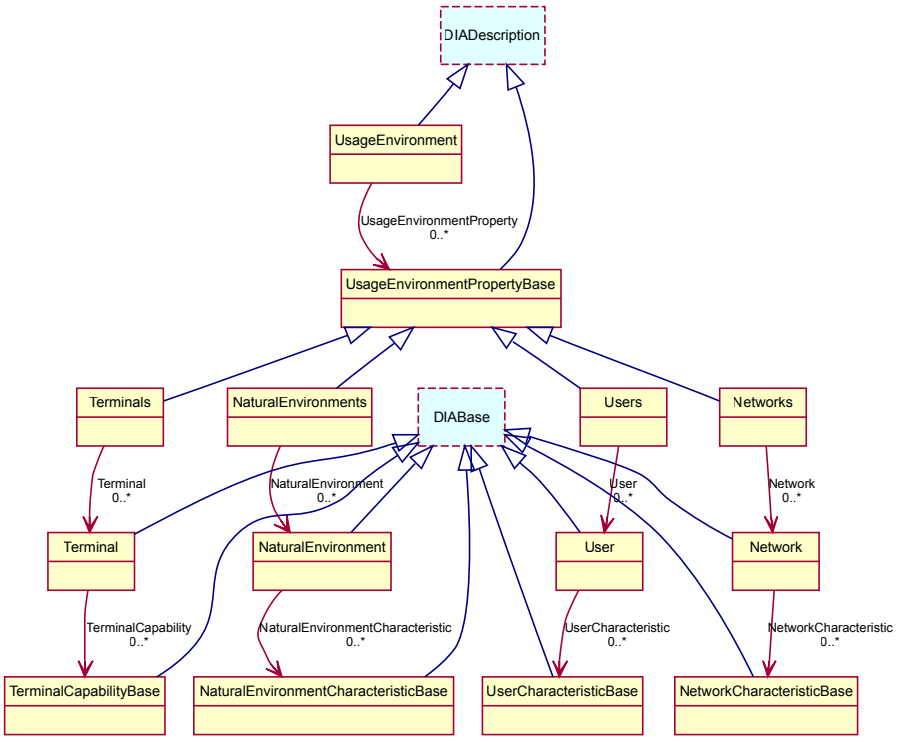


Figure A.2: MPEG-21 DIA-UED Class Model – Usage Environment.

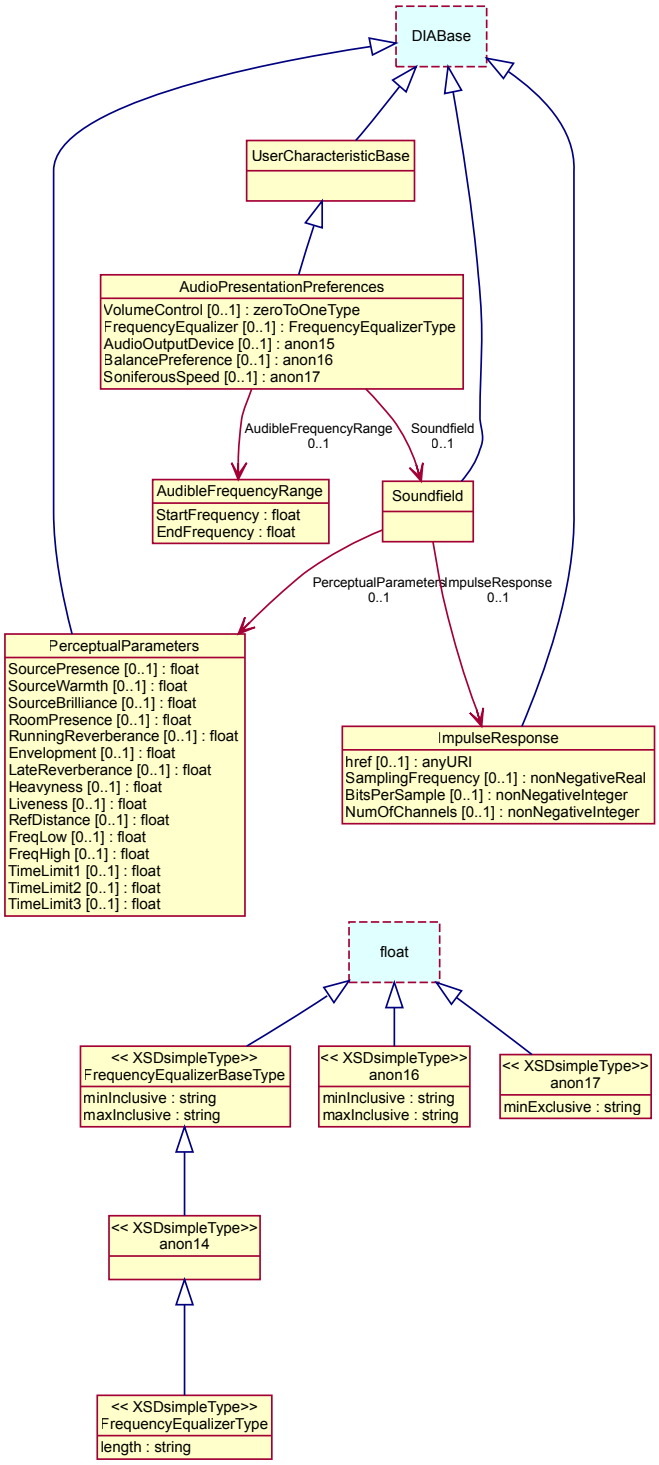


Figure A.3: MPEG-21 DIA-UED Class Model – User Characteristic (part A).

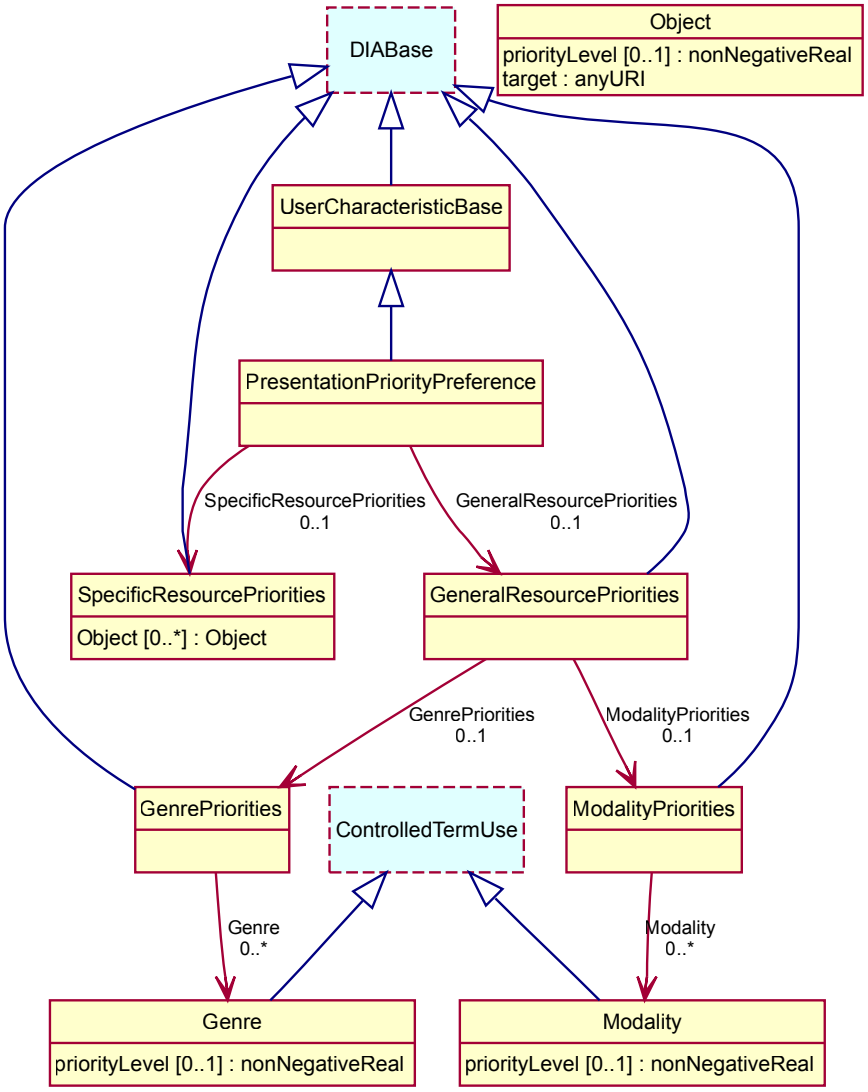


Figure A.4: MPEG-21 DIA-UED Class Model – User Characteristic (part B).

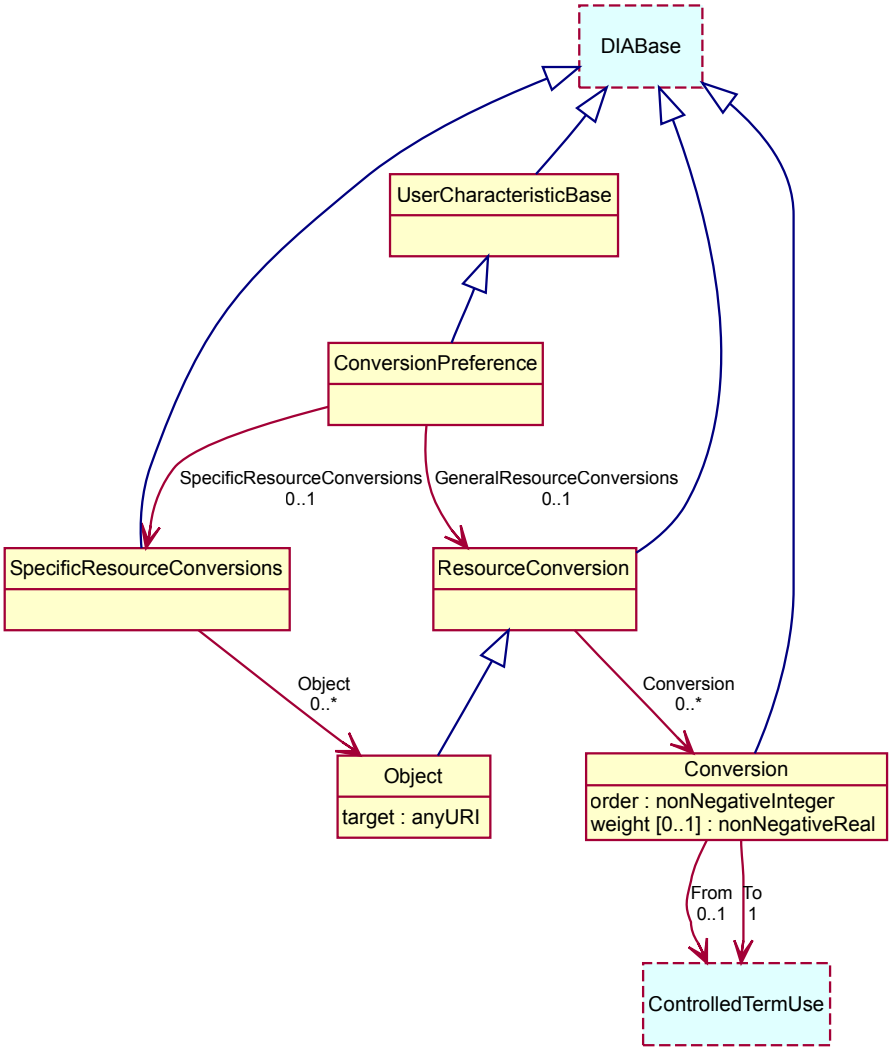


Figure A.5: MPEG-21 DIA-UED Class Model – User Characteristic (part C).

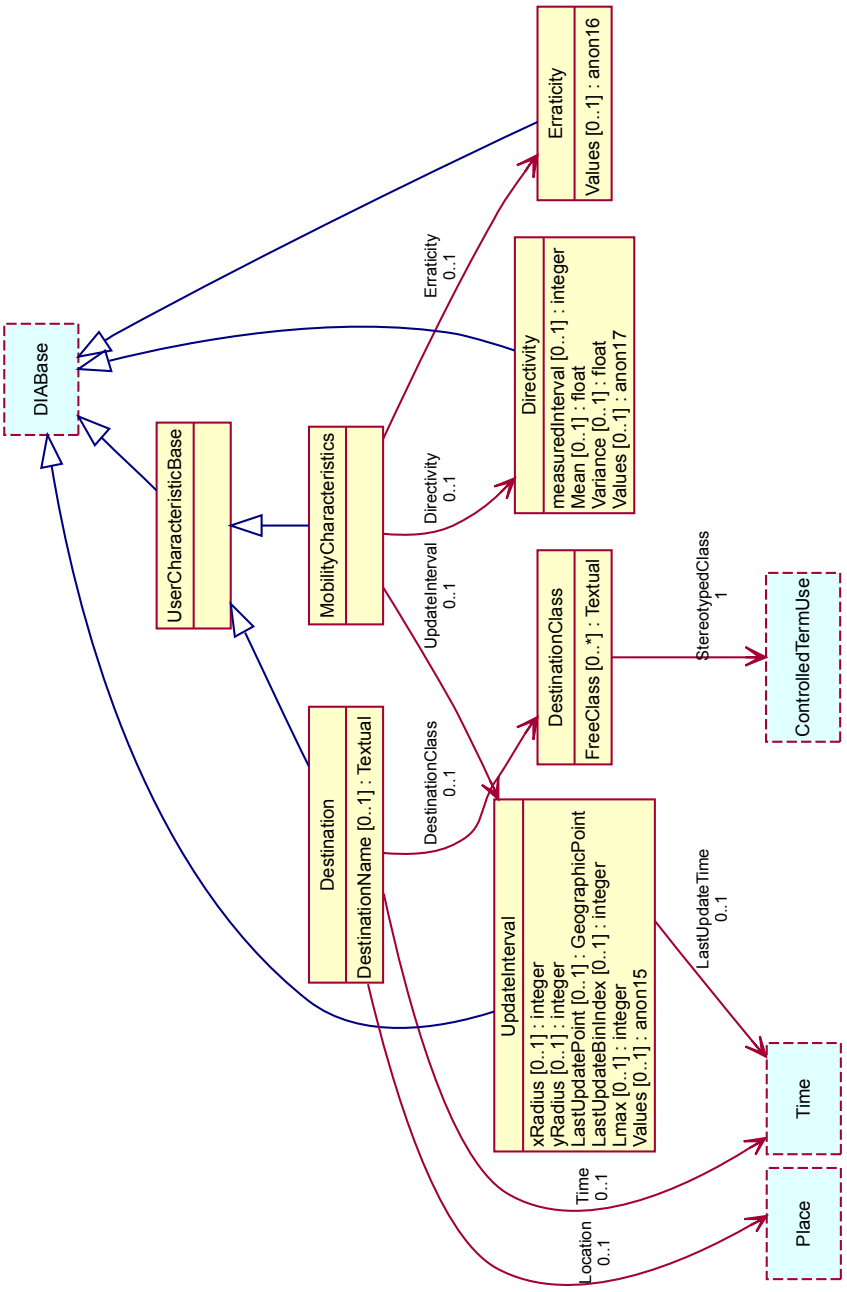


Figure A.6: MPEG-21 DIA-UED Class Model – User Characteristic (part D).

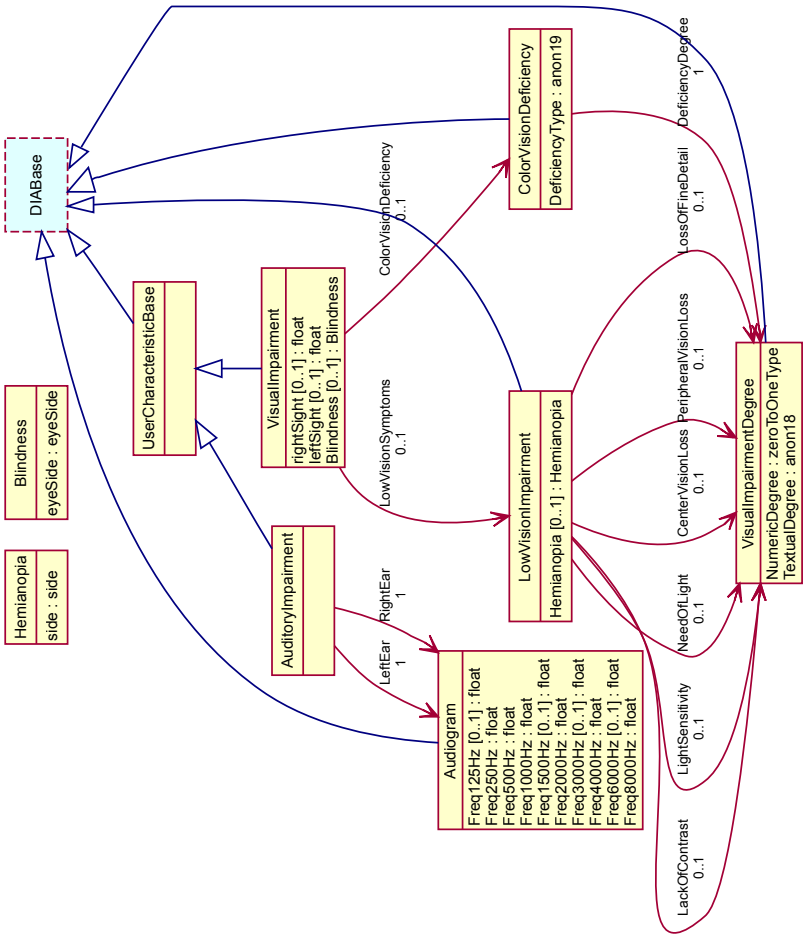


Figure A.7: MPEG-21 DIA-UED Class Model – User Characteristic (part E).

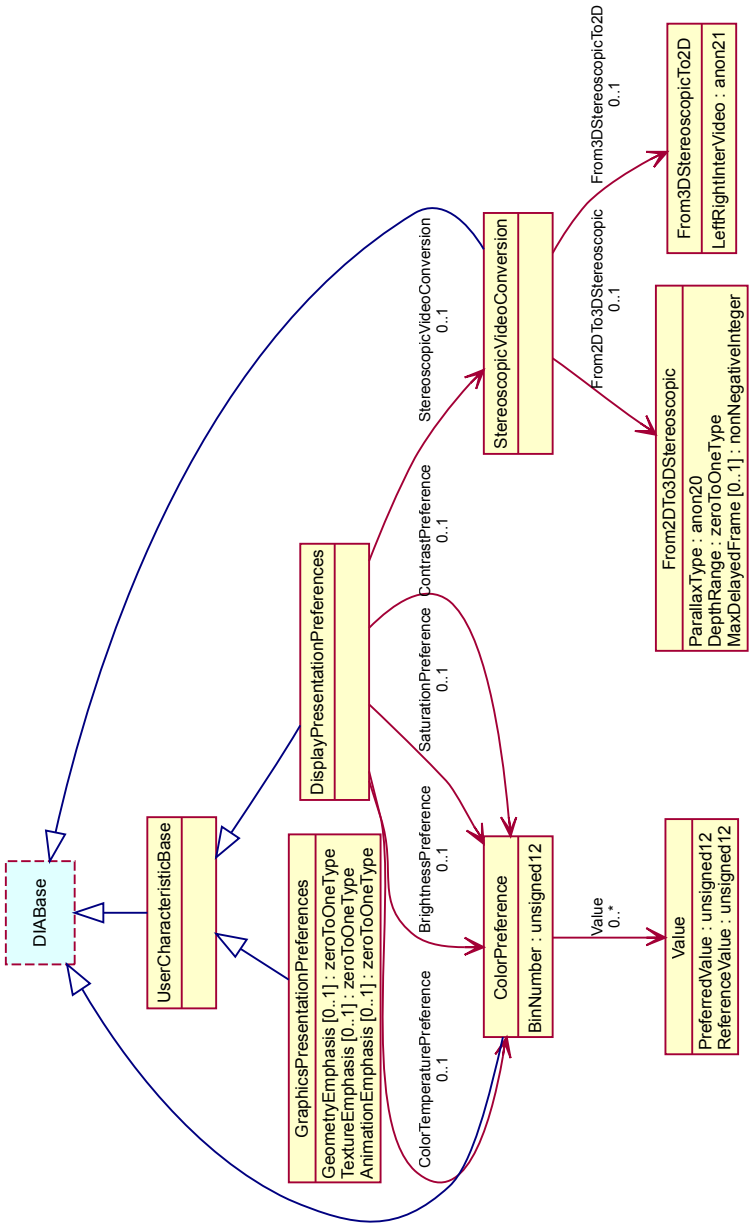


Figure A.8: MPEG-21 DIA-UED Class Model – User Characteristic (part F).

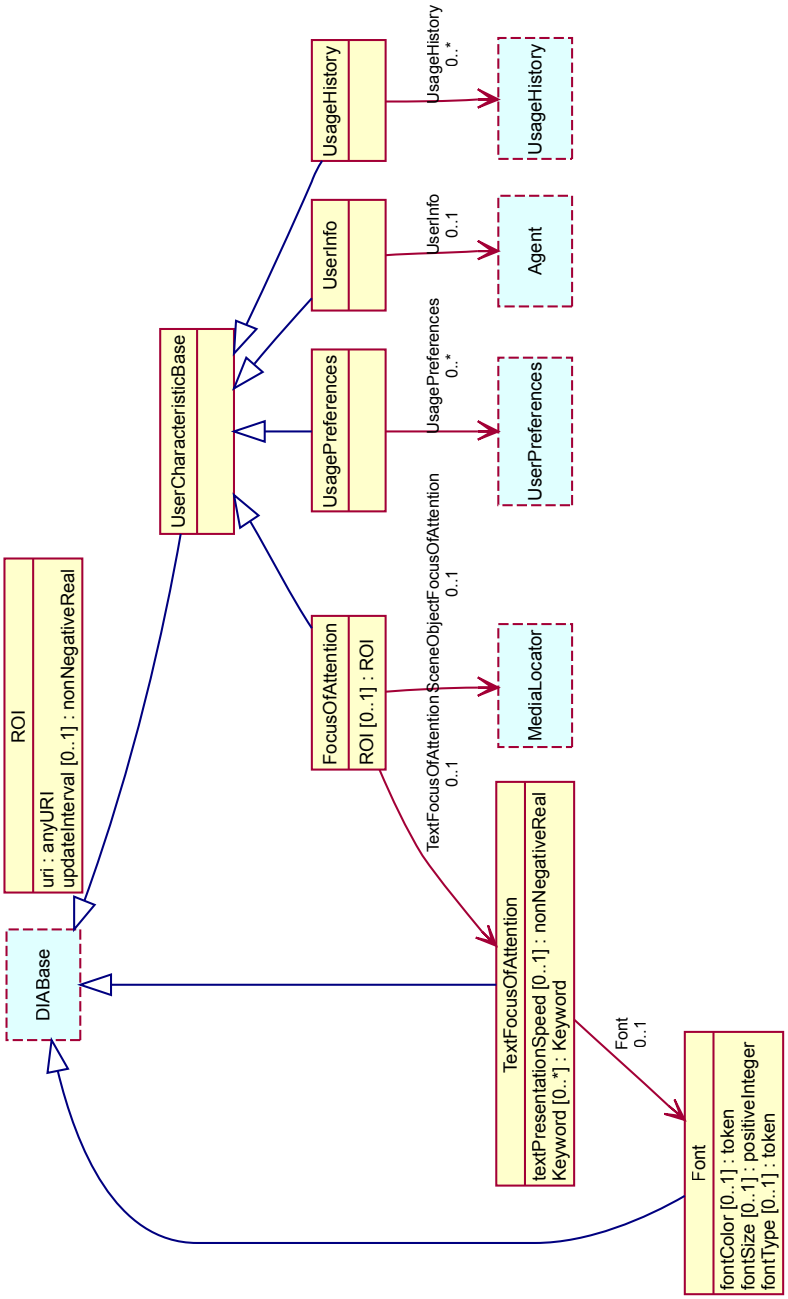


Figure A.9: MPEG-21 DIA-UED Class Model – User Characteristic (part G).

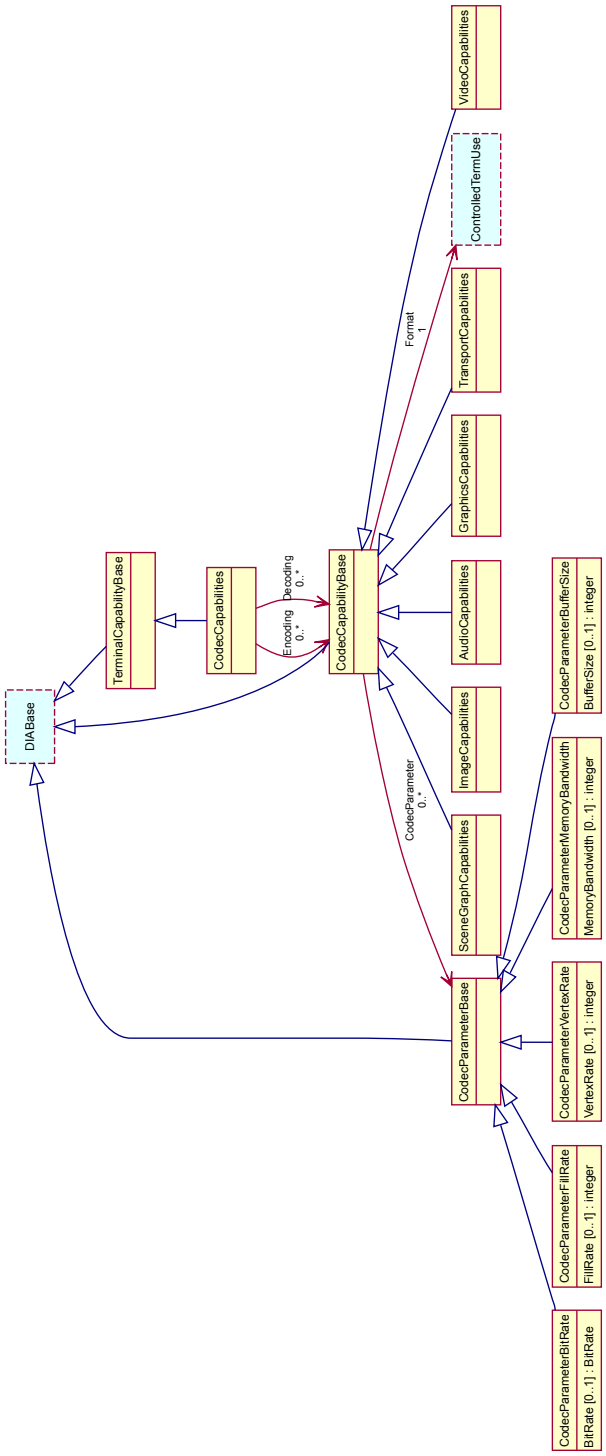


Figure A.10: MPEG-21 DIA-UED Class Model – Terminal Capability (part A).

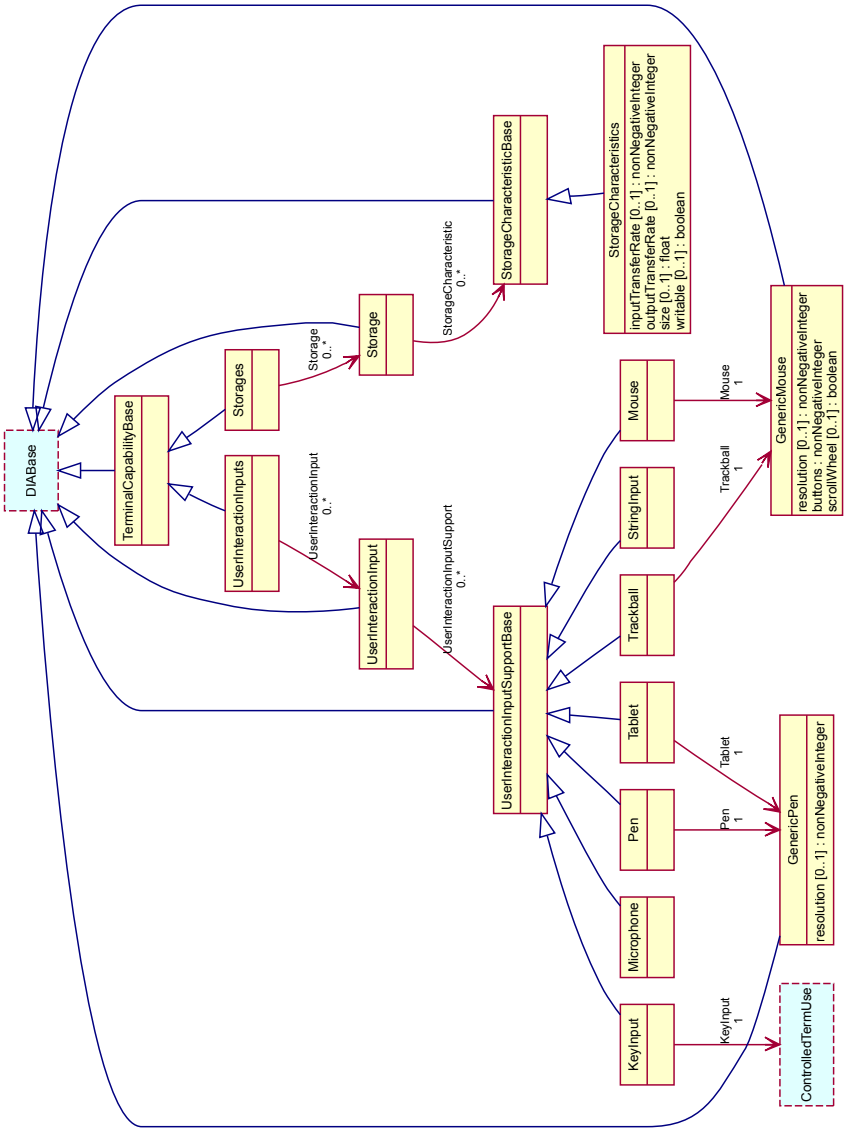


Figure A.11: MPEG-21 DIA-UED Class Model – Terminal Capability (part B).

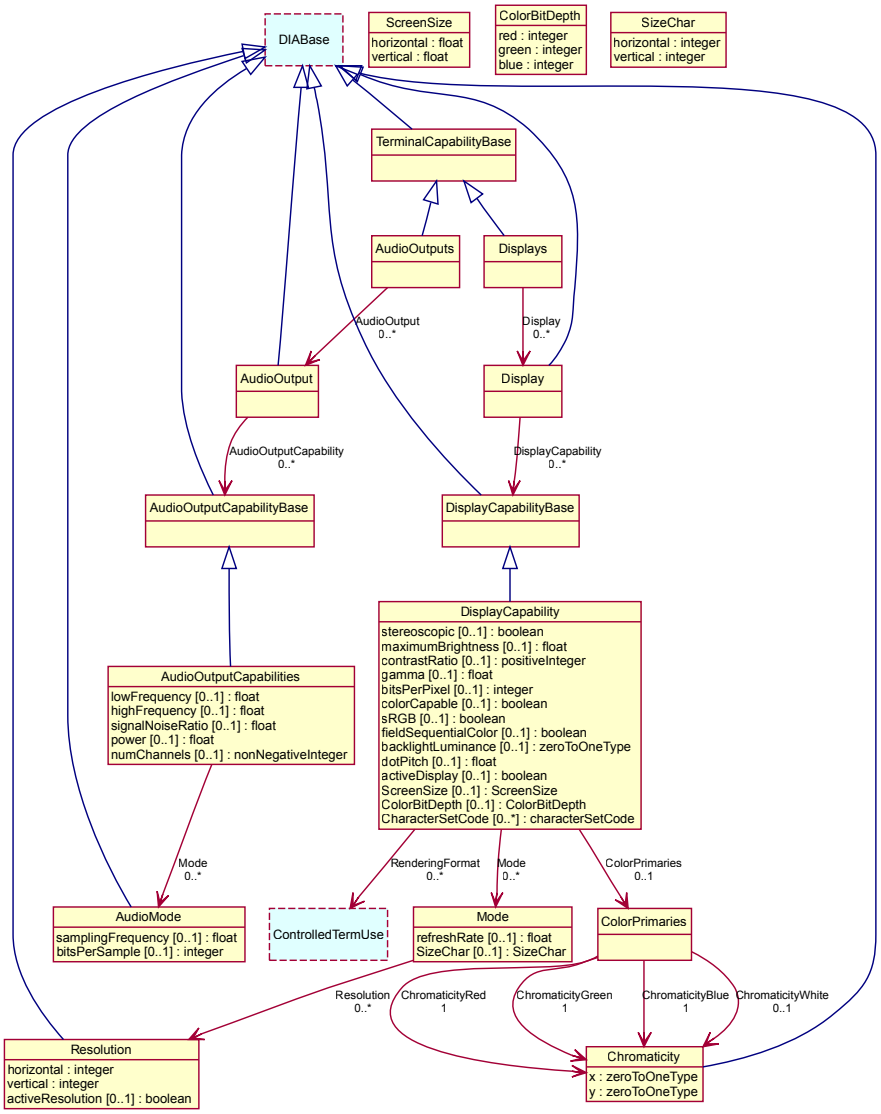


Figure A.12: MPEG-21 DIA-UED Class Model – Terminal Capability (part C).

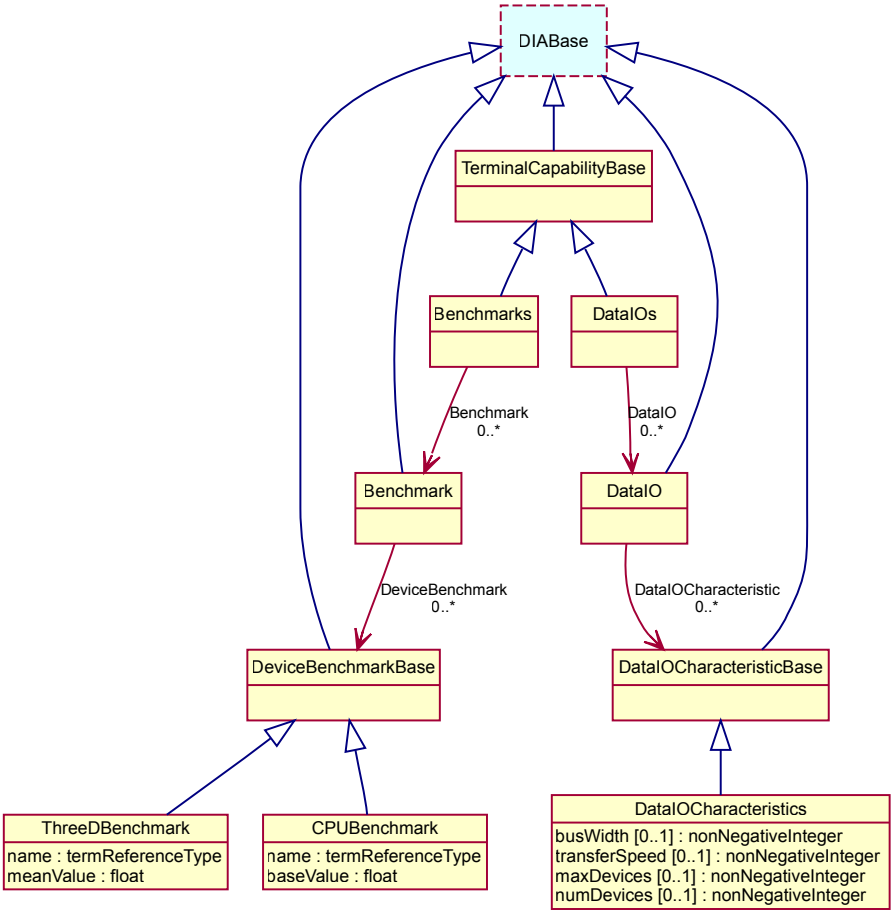


Figure A.13: MPEG-21 DIA-UED Class Model – Terminal Capability (part D).

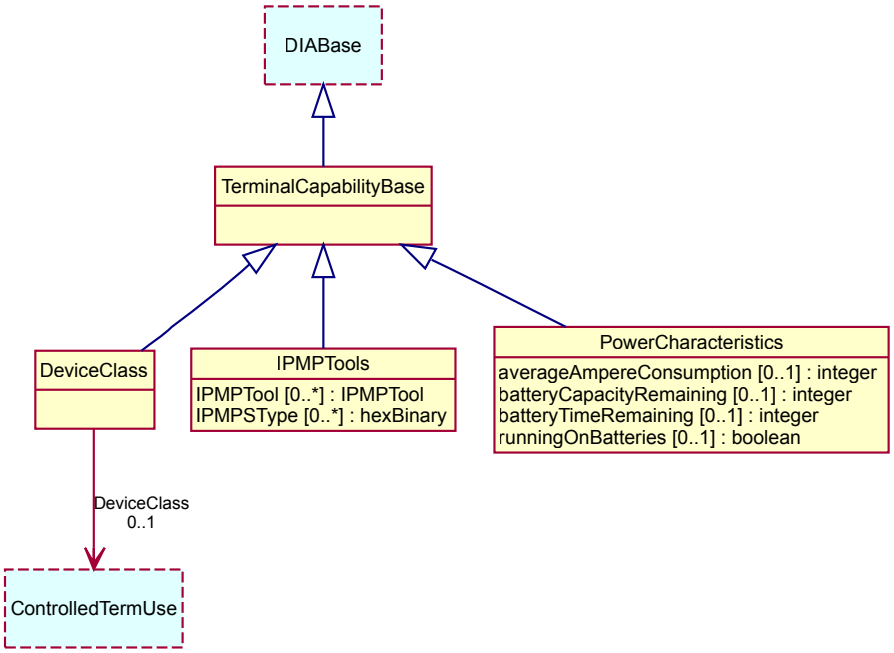


Figure A.14: MPEG-21 DIA-UED Class Model – Terminal Capability (part E).

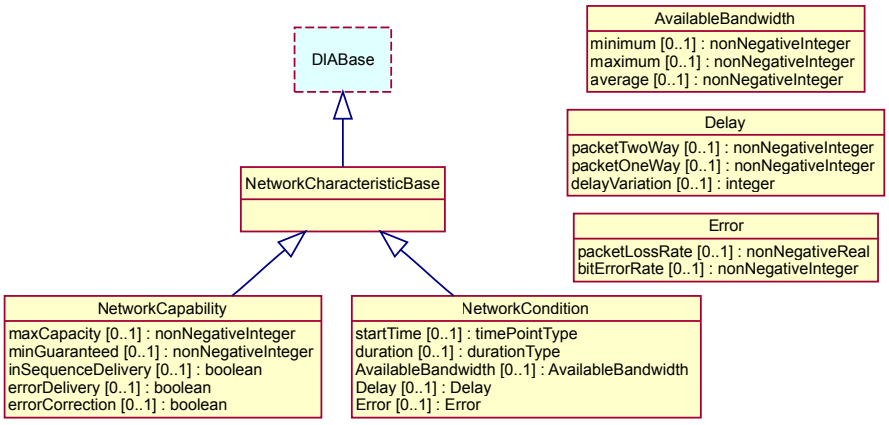


Figure A.15: MPEG-21 DIA-UED Class Model – Network Characteristic.

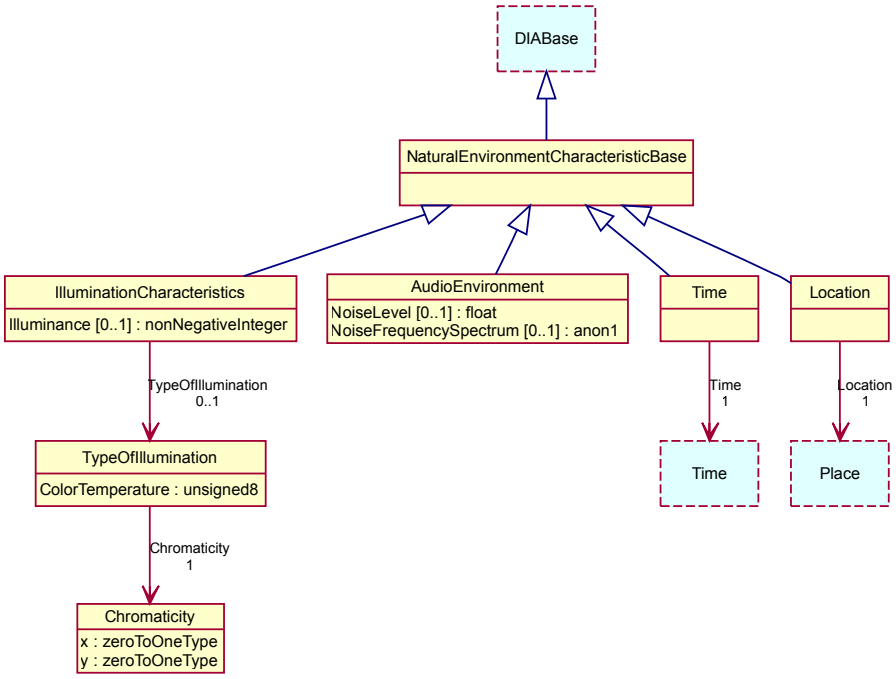


Figure A.16: MPEG-21 DIA-UED Class Model – Natural Environment Characteristic.

A.3 Examples

A.3.1 UED Complete Example 1.

Listing A.1: MPEG-21 DIA-UED example 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA
 xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
 schemaDocs/UED.xsd">
 <Description xsi:type="UsageEnvironmentType">
 <!-- User Characteristics -->
 <UsageEnvironmentProperty xsi:type="UsersType">
  <User>
  <UserCharacteristic xsi:type="UserInfoType">
   <UserInfo xsi:type="mpeg7:PersonType">
   <mpeg7:Name>
    <mpeg7:GivenName>Robbie</mpeg7:GivenName>
    <mpeg7:FamilyName>De Sutter</mpeg7:FamilyName>
   </mpeg7:Name>
   </UserInfo>
  </UserCharacteristic>
  <UserCharacteristic xsi:type="UsagePreferencesType">
   <UsagePreferences>
   <mpeg7:FilteringAndSearchPreferences>
    <mpeg7:ClassificationPreferences>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:1.6">
     <mpeg7:Name>Sports</mpeg7:Name>
    </mpeg7:Genre>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:3">
     <mpeg7:Name>Entertainment</mpeg7:Name>
    </mpeg7:Genre>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:6">
     <mpeg7:Name>Movies</mpeg7:Name>
    </mpeg7:Genre>
    </mpeg7:ClassificationPreferences>
   </mpeg7:FilteringAndSearchPreferences>
   </UsagePreferences>
  </UserCharacteristic>
  <UserCharacteristic xsi:type="UsageHistoryType">
   <UsageHistory>
   <mpeg7:UserActionHistory>
    <mpeg7:ObservationPeriod>
    <mpeg7:TimePoint>2000-10-09T18:00-08:00</mpeg7:TimePoint>
    <mpeg7:Duration>PT6H</mpeg7:Duration>
    </mpeg7:ObservationPeriod>
    <mpeg7:UserActionList>
    <mpeg7:ActionType>
     <mpeg7:Name>PlayStream</mpeg7:Name>
    </mpeg7:ActionType>
    <mpeg7:UserAction>
     <mpeg7:ProgramIdentifier>
     urn:mymedia:av:02-mnf-109
     </mpeg7:ProgramIdentifier>
    </mpeg7:UserAction>
    <mpeg7:UserAction>
     <mpeg7:ProgramIdentifier>
     urn:mymedia:av:14-znn-623
     </mpeg7:ProgramIdentifier>
    </mpeg7:UserAction>
    <mpeg7:UserAction>
     <mpeg7:ProgramIdentifier>
     urn:mymedia:av:73-mov-814
     </mpeg7:ProgramIdentifier>
    </mpeg7:UserAction>
    </mpeg7:UserActionList>
   </mpeg7:UserActionHistory>
   </UsageHistory>
  </UserCharacteristic>
```

```
<UserCharacteristic xsi:type="AudioPresentationPreferencesType">
 <VolumeControl>0.85</VolumeControl>
 <FrequencyEqualizer>
 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10
 -10 -10 0 0 0 0 10 10 10 10
 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10
 </FrequencyEqualizer>
 <AudibleFrequencyRange>
 <StartFrequency>20</StartFrequency>
 <EndFrequency>20000</EndFrequency>
 </AudibleFrequencyRange>
 <Soundfield>
 <ImpulseResponse href="http://www.sac.or.kr/concertHall/hallImp.wav">
  <SamplingFrequency>44100</SamplingFrequency>
  <BitsPerSample>16</BitsPerSample>
  <NumOfChannels>1</NumOfChannels>
 </ImpulseResponse>
 </Soundfield>
 <SoniferousSpeed>0.5</SoniferousSpeed>
</UserCharacteristic>
<UserCharacteristic xsi:type="MobilityCharacteristicsType">
 <UpdateInterval>
 <LastUpdatePoint latitude="35.00" longitude="135.7"/>
 <LastUpdateBinIndex>4</LastUpdateBinIndex>
 <LastUpdateTime>
  <mpeg7:TimePoint>2002-09-20T15:22+01:00</mpeg7:TimePoint>
 </LastUpdateTime>
 <Lmax>180</Lmax>
 <Values>
  0.4 0.2 0.1 0.1 0.1 0.1 0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 </Values>
 </UpdateInterval>
 <Directivity>
 <Mean>35</Mean>
 <Variance>27</Variance>
 <Values>
  0.1 0.2 0.5 0.2 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 </Values>
 </Directivity>
</UserCharacteristic>
<UserCharacteristic xsi:type="DestinationType">
 <Time><mpeg7:TimePoint>2002-09-20T17:00+01:00</mpeg7:TimePoint></Time>
 <DestinationName>Awaji Yumebutai</DestinationName>
</UserCharacteristic>
</User>
</UsageEnvironmentProperty>
<!-- Terminal Capabilities -->
<UsageEnvironmentProperty xsi:type="TerminalsType">
 <Terminal>
 <TerminalCapability xsi:type="CodecCapabilitiesType">
  <Decoding xsi:type="AudioCapabilitiesType">
  <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
   <mpeg7:Name>MPEG-1 Layer III + MPEG-2 Low Sampling Rate Layer III III</
       mpeg7:Name>
  </Format>
  <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.2">
   <mpeg7:Name>MPEG-2 Audio AAC Main Profile</mpeg7:Name>
  </Format>
  </Decoding>
  <Decoding xsi:type="ImageCapabilitiesType">
  <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
   <mpeg7:Name>JPEG</mpeg7:Name>
  </Format>
  </Decoding>
  <Decoding xsi:type="ImageCapabilitiesType">
  <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:6.1.2">
   <mpeg7:Name>JPEG2000 jp2p Profile - Level 1</mpeg7:Name>
  </Format>
  </Decoding>
  <Decoding xsi:type="VideoCapabilitiesType">
  <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.1.1">
   <mpeg7:Name>MPEG-2 Video Simple Profile @ Main Level</mpeg7:Name>
  </Format>
```

```
  </Decoding>
  <Decoding xsi:type="VideoCapabilitiesType">
  <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
   <mpeg7:Name>MPEG-4 Visual Simple Profile @ Level 1</mpeg7:Name>
  </Format>
  </Decoding>
  <Decoding xsi:type="VideoCapabilitiesType">
  <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.3.2">
   <mpeg7:Name>MPEG-4 Visual Advanced Simple Profile @ Level 1</mpeg7:Name>
  </Format>
  </Decoding>
 </TerminalCapability>
 <TerminalCapability xsi:type="DisplaysType">
  <Display id="primary_display">
  <DisplayCapability xsi:type="DisplayCapabilityType" colorCapable="true"
       activeDisplay="true">
   <Mode>
   <Resolution horizontal="240" vertical="320"/>
   </Mode>
  </DisplayCapability>
  </Display>
  <Display id="secondary_display">
  <DisplayCapability xsi:type="DisplayCapabilityType" colorCapable="false">
   <Mode>
   <Resolution horizontal="176" vertical="144"/>
   </Mode>
  </DisplayCapability>
  </Display>
 </TerminalCapability>
 <TerminalCapability xsi:type="PowerCharacteristicsType"
      batteryTimeRemaining="4200" runningOnBatteries="true"/>
 <TerminalCapability xsi:type="StoragesType">
  <Storage xsi:type="StorageType">
  <StorageCharacteristic xsi:type="StorageCharacteristicsType"
       inputTransferRate="8" size="1200" writable="true"/>
  </Storage>
 </TerminalCapability>
 <TerminalCapability xsi:type="DataIOsType">
  <DataIO xsi:type="DataIOType">
  <DataIOCharacteristic xsi:type="DataIOCharacteristicsType" busWidth="128"/
       >
  </DataIO>
 </TerminalCapability>
 </Terminal>
</UsageEnvironmentProperty>
<!-- Network Characteristics -->
<UsageEnvironmentProperty xsi:type="NetworksType">
 <Network>
 <NetworkCharacteristic xsi:type="NetworkCapabilityType" maxCapacity="384000
      " minGuaranteed="32000"/>
 <NetworkCharacteristic xsi:type="NetworkConditionType" duration="
      PT330N1000F">
  <AvailableBandwidth maximum="256000" average="80000"/>
  <Delay packetTwoWay="330" delayVariation="66"/>
  <Error packetLossRate="0.05"/>
 </NetworkCharacteristic>
 </Network>
</UsageEnvironmentProperty>
<!-- Natural Envrionment Characteristics -->
<UsageEnvironmentProperty xsi:type="NaturalEnvironmentsType">
 <NaturalEnvironment>
 <NaturalEnvironmentCharacteristic xsi:type="LocationType">
  <Location>
  <mpeg7:GeographicPosition>
   <mpeg7:Point longitude="135.75" latitude="35.00" altitude="10.00"/>
  </mpeg7:GeographicPosition>
  <mpeg7:Region>jp</mpeg7:Region>
  </Location>
 </NaturalEnvironmentCharacteristic>
 <NaturalEnvironmentCharacteristic xsi:type="TimeType">
  <Time>
  <mpeg7:TimePoint>1998-07-10T15:22+01:00</mpeg7:TimePoint>
  </Time>
 </NaturalEnvironmentCharacteristic>
 <NaturalEnvironmentCharacteristic xsi:type="AudioEnvironmentType">
  <NoiseLevel>20</NoiseLevel>
  <NoiseFrequencySpectrum>
```

```
    40 30 20 10 10 10 10 10 10 10
    10 40 40 40 30 30 30 20 20 20
    10 10 10 10 10 10 10 10 10 10
    10 10 10
   </NoiseFrequencySpectrum>
  </NaturalEnvironmentCharacteristic>
  </NaturalEnvironment>
 </UsageEnvironmentProperty>
 </Description>
</DIA>
```

A.3.2 UED Network Information Example.

Listing A.2: MPEG-21 DIA-UED Network Information Example.

```
<?xml version="1.0" encoding="UTF-8"?> <DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
schemaDocs/UED.xsd">
  <Description xsi:type="UsageEnvironmentType">
    <UsageEnvironmentProperty xsi:type="NetworksType">
      <Network>
        <NetworkCharacteristic xsi:type="NetworkConditionType" duration="
            PT330N1000F">
          <AvailableBandwidth maximum="9600" average="4400"/>
          <Delay packetTwoWay="330" delayVariation="66"/>
          <Error packetLossRate="0.05"/>
        </NetworkCharacteristic>
      </Network>
    </UsageEnvironmentProperty>
  </Description>
</DIA>
```

A.3.3 UED Terminal Information Example.

Listing A.3: MPEG-21 DIA-UED Terminal Information Example.

```
<?xml version="1.0" encoding="UTF-8"?> <DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
schemaDocs/UED.xsd">
  <Description xsi:type="UsageEnvironmentType">
    <UsageEnvironmentProperty xsi:type="TerminalsType">
      <Terminal>
        <TerminalCapability xsi:type="DisplaysType">
          <Display id="primary_display">
            <DisplayCapability xsi:type="DisplayCapabilityType" colorCapable="
                true" activeDisplay="true">
              <Mode>
                <Resolution horizontal="1600" vertical="1200" id="myNewScreen" /
                    >
              </Mode>
            </DisplayCapability>
          </Display>
        </TerminalCapability>
```

```
      </Terminal>
    </UsageEnvironmentProperty>
  </Description>
</DIA>
```

A.3.4 UED Complete Example 2.

Listing A.4: MPEG-21 DIA-UED Complete Example 2.

```
<?xml version="1.0" encoding="UTF-8"?> <DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
schemaDocs/UED.xsd">
 <Description xsi:type="UsageEnvironmentType">
 <!-- User Characteristics -->
 <UsageEnvironmentProperty xsi:type="UsersType">
  <User>
  <UserCharacteristic xsi:type="UserInfoType">
   <UserInfo xsi:type="mpeg7:PersonType">
   <mpeg7:Name>
    <mpeg7:GivenName>Somebody</mpeg7:GivenName>
    <mpeg7:FamilyName>Else</mpeg7:FamilyName>
   </mpeg7:Name>
   </UserInfo>
  </UserCharacteristic>
  <UserCharacteristic xsi:type="UsagePreferencesType">
   <UsagePreferences>
   <mpeg7:FilteringAndSearchPreferences>
    <mpeg7:ClassificationPreferences>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:5">
     <mpeg7:Name>Music</mpeg7:Name>
    </mpeg7:Genre>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:1.2">
     <mpeg7:Name>Drama</mpeg7:Name>
    </mpeg7:Genre>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:2">
     <mpeg7:Name>News</mpeg7:Name>
    </mpeg7:Genre>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:1.8">
     <mpeg7:Name>Politics</mpeg7:Name>
    </mpeg7:Genre>
    <mpeg7:Genre href="urn:mpeg:mpeg7:cs:GenreCS:2001:4">
     <mpeg7:Name>Society</mpeg7:Name>
    </mpeg7:Genre>
    </mpeg7:ClassificationPreferences>
   </mpeg7:FilteringAndSearchPreferences>
   </UsagePreferences>
  </UserCharacteristic>
  <UserCharacteristic xsi:type="UsageHistoryType">
   <UsageHistory>
   <mpeg7:UserActionHistory>
    <mpeg7:ObservationPeriod>
    <mpeg7:TimePoint>2004-09-09T18:00-08:00</mpeg7:TimePoint>
    <mpeg7:Duration>PT10H</mpeg7:Duration>
    </mpeg7:ObservationPeriod>
    <mpeg7:UserActionList>
    <mpeg7:ActionType>
     <mpeg7:Name>PlayStream</mpeg7:Name>
    </mpeg7:ActionType>
    <mpeg7:UserAction>
     <mpeg7:ProgramIdentifier>
      urn:elsemedia:av:15-pol-news-245
     </mpeg7:ProgramIdentifier>
    </mpeg7:UserAction>
    <mpeg7:UserAction>
     <mpeg7:ProgramIdentifier>
      urn:elsemedia:audio:75d-ds45-qq
```

```
    </mpeg7:ProgramIdentifier>
   </mpeg7:UserAction>
   <mpeg7:UserAction>
    <mpeg7:ProgramIdentifier>
    urn:elsemedia:av:46-avi-102
    </mpeg7:ProgramIdentifier>
   </mpeg7:UserAction>
   </mpeg7:UserActionList>
  </mpeg7:UserActionHistory>
  </UsageHistory>
</UserCharacteristic>
<UserCharacteristic xsi:type="AudioPresentationPreferencesType">
 <VolumeControl>0.5</VolumeControl>
 <AudibleFrequencyRange>
 <StartFrequency>40</StartFrequency>
 <EndFrequency>10000</EndFrequency>
 </AudibleFrequencyRange>
 <Soundfield>
 <ImpulseResponse href="http://mmlab.be/else/music.wma">
  <SamplingFrequency>22100</SamplingFrequency>
  <BitsPerSample>32</BitsPerSample>
  <NumOfChannels>6</NumOfChannels>
 </ImpulseResponse>
 </Soundfield>
 <SoniferousSpeed>0.7</SoniferousSpeed>
</UserCharacteristic>
<UserCharacteristic xsi:type="DisplayPresentationPreferencesType">
 <ColorTemperaturePreference>
 <BinNumber>1000</BinNumber>
 <Value>
  <PreferredValue>995</PreferredValue>
  <ReferenceValue>990</ReferenceValue>
 </Value>
 <Value>
  <PreferredValue>1995</PreferredValue>
  <ReferenceValue>1990</ReferenceValue>
 </Value>
 <Value>
  <PreferredValue>2995</PreferredValue>
  <ReferenceValue>2990</ReferenceValue>
 </Value>
 </ColorTemperaturePreference>
 <BrightnessPreference>
 <BinNumber>606</BinNumber>
 <Value>
  <PreferredValue>105</PreferredValue>
  <ReferenceValue>190</ReferenceValue>
 </Value>
 <Value>
  <PreferredValue>456</PreferredValue>
  <ReferenceValue>159</ReferenceValue>
 </Value>
 <Value>
  <PreferredValue>798</PreferredValue>
  <ReferenceValue>165</ReferenceValue>
 </Value>
 </BrightnessPreference>
 <SaturationPreference>
 <BinNumber>606</BinNumber>
 <Value>
  <PreferredValue>105</PreferredValue>
  <ReferenceValue>190</ReferenceValue>
 </Value>
 <Value>
  <PreferredValue>456</PreferredValue>
  <ReferenceValue>159</ReferenceValue>
 </Value>
 </SaturationPreference>
 <ContrastPreference>
 <BinNumber>606</BinNumber>
 <Value>
  <PreferredValue>15</PreferredValue>
  <ReferenceValue>20</ReferenceValue>
 </Value>
 <Value>
  <PreferredValue>25</PreferredValue>
  <ReferenceValue>36</ReferenceValue>
```

```
  </Value>
  <Value>
   <PreferredValue>50</PreferredValue>
   <ReferenceValue>60</ReferenceValue>
  </Value>
  <Value>
   <PreferredValue>100</PreferredValue>
   <ReferenceValue>160</ReferenceValue>
  </Value>
  <Value>
   <PreferredValue>800</PreferredValue>
   <ReferenceValue>1000</ReferenceValue>
  </Value>
  <Value>
   <PreferredValue>2000</PreferredValue>
   <ReferenceValue>2600</ReferenceValue>
  </Value>
  </ContrastPreference>
  <StereoscopicVideoConversion>
  <From2DTo3DStereoscopic>
   <ParallaxType>Positive</ParallaxType>
   <DepthRange>0.7</DepthRange>
   <MaxDelayedFrame>3</MaxDelayedFrame>
  </From2DTo3DStereoscopic>
  <From3DStereoscopicTo2D>
   <LeftRightInterVideo>Right</LeftRightInterVideo>
  </From3DStereoscopicTo2D>
  </StereoscopicVideoConversion>
 </UserCharacteristic>
 <UserCharacteristic xsi:type="GraphicsPresentationPreferencesType">
  <GeometryEmphasis>1.0</GeometryEmphasis>
  <TextureEmphasis>0.5</TextureEmphasis>
  <AnimationEmphasis>0.5</AnimationEmphasis>
 </UserCharacteristic>
 <UserCharacteristic xsi:type="ConversionPreferenceType">
  <GeneralResourceConversions>
  <Conversion order="1" weight="1.0">
   <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
   <mpeg7:Name>Video</mpeg7:Name>
   </From>
   <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
   <mpeg7:Name>Video</mpeg7:Name>
   </To>
  </Conversion>
  <Conversion order="3" weight="1.0">
   <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
   <mpeg7:Name>Video</mpeg7:Name>
   </From>
   <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.1">
   <mpeg7:Name>Image</mpeg7:Name>
   </To>
  </Conversion>
  <Conversion order="2" weight="1.0">
   <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
   <mpeg7:Name>Video</mpeg7:Name>
   </From>
   <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
   <mpeg7:Name>Audio</mpeg7:Name>
   </To>
  </Conversion>
  <Conversion order="4" weight="1.0">
   <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
   <mpeg7:Name>Video</mpeg7:Name>
   </From>
   <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:5">
   <mpeg7:Name>Text</mpeg7:Name>
   </To>
  </Conversion>
  </GeneralResourceConversions>
 </UserCharacteristic>
 <UserCharacteristic xsi:type="PresentationPriorityPreferenceType">
  <GeneralResourcePriorities>
  <ModalityPriorities>
   <Modality priorityLevel="1.5"
    href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
   <mpeg7:Name>Video</mpeg7:Name>
   </Modality>
```

```
    </ModalityPriorities>
    <GenrePriorities>
     <Genre priorityLevel="1.6"
     href="urn:mpeg:mpeg7:cs:GenreCS:2001:1.6">
     <mpeg7:Name>Sports</mpeg7:Name>
     </Genre>
    </GenrePriorities>
    </GeneralResourcePriorities>
   </UserCharacteristic>
   <UserCharacteristic xsi:type="MobilityCharacteristicsType">
    <UpdateInterval>
    <LastUpdatePoint latitude="45.05" longitude="105.0"/>
    <LastUpdateBinIndex>4</LastUpdateBinIndex>
    <LastUpdateTime>
     <mpeg7:TimePoint>2004-09-10T12:00+01:00</mpeg7:TimePoint>
    </LastUpdateTime>
    <Lmax>150</Lmax>
    <Values>
     0.4 0.2 0.1 0.1 0.1 0.1 0.0 0.0
     0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
     0.0 0.0 0.0 1.0 0.4 0.5 0.0 0.0
     0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    </Values>
    </UpdateInterval>
    <Directivity>
    <Mean>15</Mean>
    <Variance>20</Variance>
    <Values>
     0.1 0.2 0.5 0.2 0.0 0.0 0.0 0.0
     0.0 1.0 0.0 0.8 0.5 0.5 0.4 0.0
    </Values>
    </Directivity>
   </UserCharacteristic>
   <UserCharacteristic xsi:type="DestinationType">
    <Time>
    <mpeg7:TimePoint>2004-09-09T18:00+01:00</mpeg7:TimePoint>
    </Time>
    <DestinationName>Ghent</DestinationName>
   </UserCharacteristic>
   </User>
  </UsageEnvironmentProperty>
  <!-- Natural Envrionment Characteristics -->
  <UsageEnvironmentProperty xsi:type="NaturalEnvironmentsType">
   <NaturalEnvironment>
   <NaturalEnvironmentCharacteristic xsi:type="LocationType">
    <Location>
    <mpeg7:GeographicPosition>
     <mpeg7:Point longitude="14.20" latitude="46.39" altitude="448.00"/>
    </mpeg7:GeographicPosition>
    <mpeg7:Region>at</mpeg7:Region>
    </Location>
   </NaturalEnvironmentCharacteristic>
   <NaturalEnvironmentCharacteristic xsi:type="TimeType">
    <Time>
    <mpeg7:TimePoint>2004-09-13T17:33+01:00</mpeg7:TimePoint>
    </Time>
   </NaturalEnvironmentCharacteristic>
   <NaturalEnvironmentCharacteristic xsi:type="AudioEnvironmentType">
    <NoiseLevel>40</NoiseLevel>
    <NoiseFrequencySpectrum>
    10 10 10 10 10 10 10 10 10 10
    10 10 10 10 30 30 10 10 10 10
    10 10 10 10 10 10 10 10 10 10
    10 10 10
    </NoiseFrequencySpectrum>
   </NaturalEnvironmentCharacteristic>
   <NaturalEnvironmentCharacteristic
    xsi:type="IlluminationCharacteristicsType">
    <TypeOfIllumination>
    <ColorTemperature>105</ColorTemperature>
    </TypeOfIllumination>
    <Illuminance>250</Illuminance>
   </NaturalEnvironmentCharacteristic>
   </NaturalEnvironment>
  </UsageEnvironmentProperty>
  </Description>
</DIA>
```

A.4 MPEG-21 DIA-UED Software Toolkit Usage

Listing A.5: UED Toolkit test – API Usage (excerpt).

```
// prefix the first occurrence of StartFrequency with "10".
String theStartFreq[] = (theAFR[0].getStartFrequency());
theStartFreq[0] = new String("10").concat(theStartFreq[0]);
theAFR[0].setStartFrequency(theStartFreq);

// set the first occurrence of SoniferousSpeed to 1.9
String theSoniferousSpeed[] = new String[1];
theSoniferousSpeed[0] = new String("1.9");
theAPPT.setSoniferousSpeed(theSoniferousSpeed);

// create a new element: BalancedPreferred
String balancePreference[] = new String[1];
balancePreference[0] = new String("-5");
theAPPT.setBalancePreference(balancePreference);

// delete the element Soundfield
theAPPT.setSoundfield(new SoundfieldType[0]);

// network
NetworkType theNetwork = theNetworks.getNetwork()[0];
NetworkCharacteristic[] currentNC = theNetwork.
    getNetworkCharacteristics();

// add a new networkCharacteristic
NetworkCharacteristic newNC = new NetworkCharacteristic();

NetworkConditionType newNCC = new NetworkConditionType();
newNCC.setDuration("PT500N2000F");

// AvailableBandwidth
AvailableBandwidth[] ab = new AvailableBandwidth[1];
ab[0] = new AvailableBandwidth();
ab[0].setMaxium(new Integer(10000));
ab[0].setMinimum(new Integer(5000));

newNCC.setAvailableBandwidth(ab);
```

Appendix B

Fast Object Tracking Algorithms Pseudo-Code

This appendix contains the pseudo-code listings for the fast object tracking algorithms introduced in Section 5.4.

Algorithm B.1: Fast object tracking algorithm – main procedure.

for *each frame* **do**
    `// Start procedure`
    `// Initialization frame`
    Algorithm B.3
    `OMV` = calculate`OMV` using Algorithm B.4
    `// Translated mask`
    $L_X = L_X + \texttt{OMV}^1$
    $L_Y = L_Y + \texttt{OMV}^2$
    `// Perform Resize Actions`
    M_H = calculate Formula 5.3
    M_V = calculate Formula 5.4
    Algorithm B.6 `// resize horizontal`
    Algorithm B.7 `// resize vertical`
    `// Determine Macroblock Shifting Values`
    Algorithm B.2
end

Algorithm B.2: Determine shifting values for all macroblocks.

```
// Reset the shifting values for all macroblocks
for i = 0 to M − 1 do
    for j = 0 to N − 1 do
        m_{i,j} = 0
    end
end
// For all matrix elements T
for x = 0 to T_C − 1 do
    for y = 0 to T_R − 1 do
        // Determine i and j
        i = trunc((L_X + 8 * x)/16)
        j = trunc((L_Y + 8 * y)/16)
        // Set shifting values to macroblocks
        if (L_X + 8 * d_x) mod 16 ≤ 8 and
        (L_Y + 8 * d_y) mod 16 ≤ 8 then
            // Figure 5.11(a)
            m_{i,j} = max(m_{i,j}, T(x, y))
        else if (L_X + 8 * d_x) mod 16 > 8 and
        (L_Y + 8 * d_y) mod 16 ≤ 8 then
            // Figure 5.11(b)
            m_{i,j} = max(m_{i,j}, T(x, y))
            m_{i+1,j} = max(m_{i+1,j}, T(x, y))
        else if (L_X + 8 * d_x) mod 16 ≤ 8 and
        (L_Y + 8 * d_y) mod 16 > 8 then
            // Figure 5.11(c)
            m_{i,j} = max(m_{i,j}, T(x, y))
            m_{i,j+1} = max(m_{i,j+1}, T(x, y))
        else if (L_X + 8 * d_x) mod 16 > 8 and
        (L_Y + 8 * d_y) mod 16 > 8 then
            // Figure 5.11(d)
            m_{i,j} = max(m_{i,j}, T(x, y))
            m_{i+1,j} = max(m_{i+1,j}, T(x, y))
            m_{i,j+1} = max(m_{i,j+1}, T(x, y))
            m_{i+1,j+1} = max(m_{i+1,j+1}, T(x, y))
        end
    end
end
```

Algorithm B.3: Initialization for each frame.

// Calculate d_x, d_y
$d_x = 8 - (L_X \bmod 8)$
$d_y = 8 - (L_Y \bmod 8)$
// Calculate P_γ and $\sum_{\gamma=1}^{4} P_\gamma$
$P_1 = (d_x * d_y)/64$
$P_2 = ((8 - d_x) * d_y)/64$
$P_3 = (d_x * (8 - d_y))/64$
$P_4 = ((8 - d_x) * (8 - d_y))/64$
if $P_1 < \lambda$ **then** $P_1 = 0$
if $P_2 < \lambda$ **then** $P_2 = 0$
if $P_3 < \lambda$ **then** $P_3 = 0$
if $P_4 < \lambda$ **then** $P_4 = 0$
$P_{sum} = P_1 + P_2 + P_3 + P_4$

Algorithm B.4: Calculate OMV.

// For all matrix elements T
for $x = 0$ **to** $T_C - 1$ **do**
  for $y = 0$ **to** $T_R - 1$ **do**
    // Get $MV^1_{x,y}$ ...$MV^4_{x,y}$ using Algorithm B.5
    // Calculate formula (5.2)
    $\mathtt{OMV}_{x,y} = \frac{MV^1_{x,y} * P_1 + MV^2_{x,y} * P_2 + MV^3_{x,y} * P_3 + MV^4_{x,y} * P_4}{P_{sum}}$
    // Apply Constraint (C.2)
    if $((x = 0$ **or** $x = T_C - 1)$ **and** $T_C > 2)$ **or**
    $((y = 0$ **or** $y = T_R - 1)$ **and** $T_R > 2)$ **then**
      $\mathtt{OMV}_{x,y} = \vec{0}$
      C2Counter++
    end
  end
end
// Calculate overall object motion result.
for $x = 0$ **to** $T_C - 1$ **do**
  for $y = 0$ **to** $T_R - 1$ **do**
    $\mathtt{OMV} += \mathtt{OMV}_{x,y}$
  end
end
$\mathtt{OMV} = \mathtt{OMV}/((T_C * T_R) - \text{C2Counter})$

Algorithm B.5: Determine $MV_{x,y}$.

// Determine i and j
$i = \textbf{trunc}((L_X + 8 * x)/16)$
$j = \textbf{trunc}((L_Y + 8 * y)/16)$
// Determine $MV_{x,y}$
if $(L_X + 8 * d_x) \bmod 16 \leq 8$ **and** $(L_Y + 8 * d_y) \bmod 16 \leq 8$ **then**
    // Figure 5.12(a)
    $MV^1_{x,y} = MV^1_{i,j}$
    $MV^2_{x,y} = MV^2_{i,j}$
    $MV^3_{x,y} = MV^3_{i,j}$
    $MV^4_{x,y} = MV^4_{i,j}$
else if $(L_X + 8 * d_x) \bmod 16 > 8$ **and** $(L_Y + 8 * d_y) \bmod 16 \leq 8$ **then**
    // Figure 5.12(b)
    $MV^1_{x,y} = MV^2_{i,j}$
    $MV^2_{x,y} = MV^1_{i+1,j}$
    $MV^3_{x,y} = MV^4_{i,j}$
    $MV^4_{x,y} = MV^3_{i+1,j}$
else if $(L_X + 8 * d_x) \bmod 16 \leq 8$ **and** $(L_Y + 8 * d_y) \bmod 16 > 8$ **then**
    // Figure 5.12(c)
    $MV^1_{x,y} = MV^3_{i,j}$
    $MV^2_{x,y} = MV^4_{i,j}$
    $MV^3_{x,y} = MV^1_{i,j+1}$
    $MV^4_{x,y} = MV^2_{i,j+1}$
else if $(L_X + 8 * d_x) \bmod 16 > 8$ **and** $(L_Y + 8 * d_y) \bmod 16 > 8$ **then**
    // Figure 5.12(d)
    $MV^1_{x,y} = MV^4_{i,j}$
    $MV^2_{x,y} = MV^3_{i+1,j}$
    $MV^3_{x,y} = MV^2_{i,j+1}$
    $MV^4_{x,y} = MV^1_{i+1,j+1}$
end

Algorithm B.6: Perform Resize Actions – horizontal.

```
// Action (1.a)
if M_H > δ then
    repeat
        for x = 0 to T_C − 2 do
            for y = 0 to T_R − 1 do
                T'(x, y) = round ( ((T_C − x − 1)T(x,y) + (x + 1)T(x+1,y)) / T_C )
            end
        end
        M_H = M_H − 8
        L_X = L_X + 4
    until M_H ≤ δ
// Action (1.c)
else if M_H < −δ then
    repeat
        for x = 0 to T_C do
            for y = 0 to T_R − 1 do
                T'(x, y) = round ( ((T_C − x)T(x,y) + (x)T(x−1,y)) / T_C )
            end
        end
        M_H = M_H + 8
        L_X = L_X − 4
    until M_H ≥ −δ
end
```

Algorithm B.7: Perform Resize Actions – vertical.

```
// Action (2.a)
if M_V > δ then
    repeat
        for x = 0 to T_C − 1 do
            for y = 0 to T_R − 2 do
                T'(x, y) = round ( ((T_R − y − 1)T(x,y) + (y+1)T(x,y+1)) / T_R )
            end
        end
        M_V = M_V − 8
        L_Y = L_Y + 4
    until M_V ≤ δ
// Action (2.c)
else if M_V < −δ then
    repeat
        for x = 0 to T_C − 1 do
            for y = 0 to T_R do
                T'(x, y) = round ( ((T_R − y)T(x,y) + (y)T(x,y−1)) / T_R )
            end
        end
        M_V = M_V + 8
        L_Y = L_Y − 4
    until M_V ≥ −δ
end
```

Appendix C

UMA-compliant Video-on-Demand Application

C.1 Introduction

In this appendix, we discuss in detail how to develop a UMA-compliant Internet-based Video-on-Demand application with support for time-varying metadata as introduced in Section 6.1. We combine the results of all previous chapters and select the suitable techniques in order to realize this desired VoD application.

We have chosen to develop a VoD application as this kind of application is ideal to evaluate the usefulness of our contributions discussed in this thesis and to evaluate whether or not these new techniques can improve the existing VoD solutions. In addition, we observe that Video-on-Demand applications are increasingly being used, as illustrated by the rapidly increasing numbers of available *video blogs*.

Current Internet-based VoD systems require an end user to manually select the type of video by answering several (technical) questions, illustrated in Figure C.1. Most common questions are the preferred video encoding scheme and player (e.g., Apple's "QuickTime"[1] or Microsoft's

[1] More information on QuickTime is available at `http://www.apple.com/quicktime`.

"Windows Media Player"[2]), the network connection speed (e.g., "broadband" versus "dial-up" – in Figure C.1 denoted as "high quality" and "low quality" respectively), and the preferred video size (e.g., "small," "medium," "large," or "high definition"). This information is used to select one particular version of the video from a repository of semantically equal audio-visual streams with different technical characteristics – i.e., a *simulstore*. For each possible selection a different video stream exists.

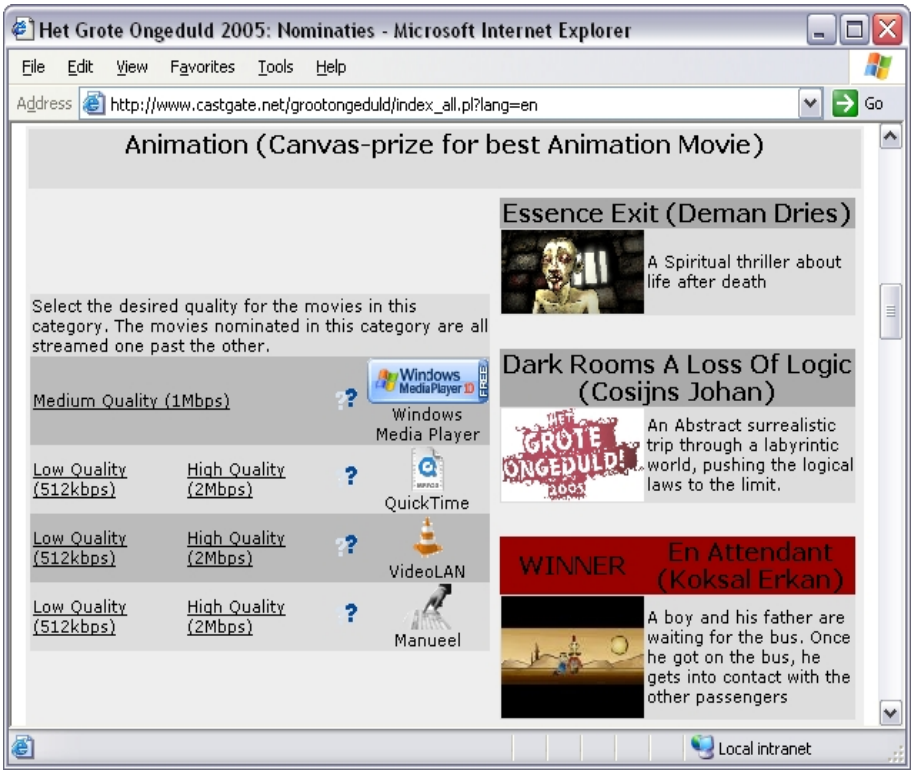


Figure C.1: Example of an actual Internet-based VoD system: the end user manually selects the type of video by answering (technical) questions.

Next, the selected video is streamed to the end-user device. If the user made a faulty choice, the content is either not consumable on his device (for example, he has chosen "QuickTime" as video player, but his device is not equipped with this player) or the stream does not fully exploit the

[2]More information on Windows Media Player is available at `http://www.microsoft.com/windows/windowsmedia`.

possibilities of the context (for example, the end user selected "low quality" although his network connection fits the bandwidth requirements for "high quality"). In addition, the stream restarts completely when the end user changes one of his choices during the consumption of the audio-visual stream, for example, he notices that "high quality" does not work well on his device, so he selects "low quality."

As one can see, the traditional Internet-based VoD systems have various limitations. Although improvements are made by the content providers, such as a reduction in the number of (technical) questions, it is still not compliant to the UMA principles. For example, the use of a simulstore results in sub-optimal content being delivered to the consumer.

In addition to the Internet-based applications, other VoD applications gain popularity, especially thanks to digital television. For example, in Flanders the "net gemist" ("just missed") services of the public broadcaster *Flemish Radio- and Television Network* (VRT)[3] allows digital television subscribers to request television programs via their set-top boxes on demand. Although this is a VoD application, we will not consider it because the context is pre-determined. Indeed, the network and the capabilities of the set-top boxes are exactly known and identical for all end users. As such, it is possible to optimize the audio-visual content to this fixed context in advance.

In this appendix, we first discuss the architecture for the intended VoD application as well as the operating procedures in the usage scenario. Next, we select the feasible technologies discussed in this thesis to handle the different tasks defined in the scenario. Finally, we construct a VoD application based on the described architecture and the selected technologies.

C.2 Architecture and Usage Scenario

As stated in the introduction, our VoD application must be UMA compliant and capable of handling time-varying metadata. We have chosen to create an architecture that closely resembles the one discussed in Section 3.4. As such, it contains four components:

- *Client*: this comprises the end user and his device on which the audio-visual content is consumed. The end-user device handles all

[3]More information on the VRT is available at `http://www.vrt.be`.

communication with the end user through a GUI.

- *Network*: an IP-based network, such as the Internet.
- *Broker*: an intermediary component that contains an inventory of the available audio-visual content. The broker also decides how to optimize the content, thus also being the content adaptation decision engine.
- *Content Provider*: this component is responsible to adapt and stream the content over the network to the client. The content adaptation engine is placed here as recommended in Section 3.2.

The Video-on-Demand mode of operation is divided in two phases: an *initialization* phase and a *consumption* phase. During the first phase, several steps are executed as illustrated in Figure C.2:

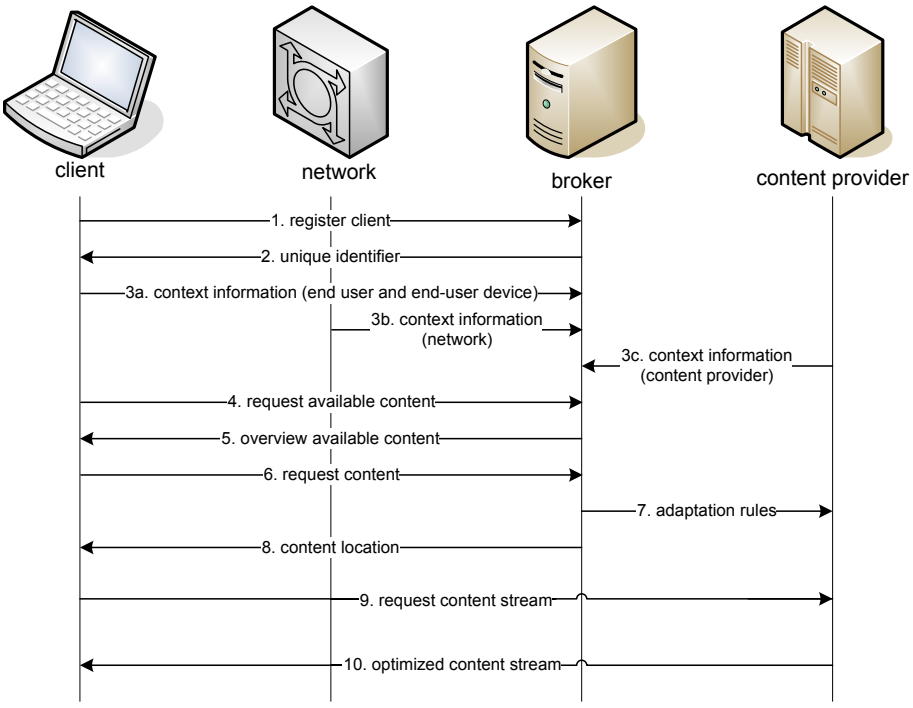


Figure C.2: Initialization of the VoD application.

1. To start, the client registers itself with the broker.

2. The broker replies with a unique identifier for this session. All subsequent communication must contain this identifier.

3. In the third step, the broker receives information about the context:

   a. The client sends information about the end user and the end-user device.

   b. The network pushes its context information, for example available bandwidth, to the broker.

   c. The content provider informs the broker about its status, for example workload and remaining processing capacity.

   It is desirable to negotiate this information at the start of the application so the broker can take the context into account for all upcoming replies.

4. The client requests an overview of the available content.

5. The broker creates the requested overview and eliminates the content that is indisputably not feasible for the given context, for example, if the quality of the adapted version is unacceptable. The client receives the list and presents it to the end user.

6. After the end user selected a video, the client informs the broker about the choice. The broker retrieves information on the selected video and the content provider. If the broker manages multiple content providers, the broker can select the best provider for the given context, for example, the one nearest to the client.

7. Having the information about the context, the chosen content, and the selected content provider, the broker determines the adaptation rules. These rules are transmitted to the content provider.

8. The broker informs the client on the location of the content.

9. The client requests the content from the content provider.

10. The content provider optimizes the requested content using the received adaptation rules from the broker. Afterwards, the content provider streams this optimized content over the network to the client.

After the initialization phase, the client receives content optimized to the initial context. This optimized content is derived from a single content base, hence, this scenario is compliant to the principles of the UMA framework.

However, during consumption, we expect that the context will change according to the concept of time-varying metadata as described in Section 3.4.2. If so, we want to re-optimize the content to the new context. This results in the following steps (Figure C.3):

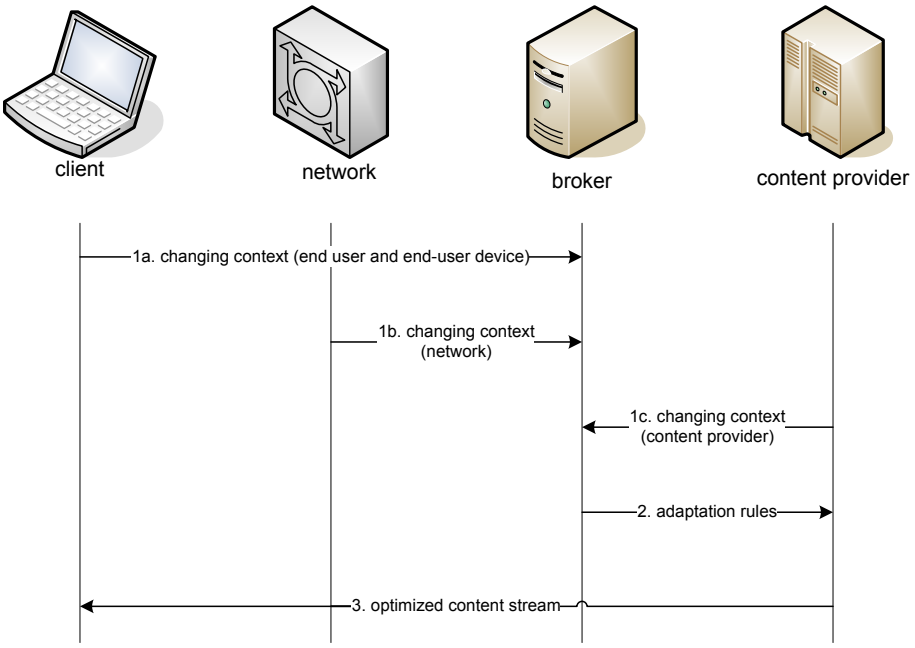


Figure C.3: Handling time-varying metadata during content consumption.

1. The broker is informed about the changed context. This can be done by either one of the following parts:
   a. The client sends information about the changed context information on the end user and the end-user device.
   b. The network sends information about the changed context information of the network capabilities and conditions.
   c. The content provider informs the broker about its changed context, e.g. current processing load.

2. The broker re-determines the adaptation rules and sends these to the content provider.

3. The content provider re-optimizes the content on-the-fly, i.e., without restarting the session.

These steps can be repeated as many times as required. As such, our scenario is compliant to the principles of the UMA framework and handles time-varying metadata.

C.3 Technologies

In this section, we assign a feasible technology to accomplish a particular step. First, we select the technologies for the steps in the initialization phase depicted in Figure C.2:

1. The registration process can be seen as invoking a web service in order to retrieve the unique session identifier. Hence, we can use any of the RPC's discussed in Section 3.3. Here, we select the SOAP over HTTP technology, mainly because it is very straightforward to use and to implement. Furthermore, the infrastructure to support SOAP – in general, the Web service infrastructure – is currently natively supported by most Web servers, which facilitates the construction.

   In order to have a uniform interaction model, SOAP is always used for any communication with the broker.

2. The broker replies to the SOAP message of step 1 with an acknowledgment of successful registration containing the unique identifier that was determined for the session.

3. In the third step of the initialization phase, the client, the network, and the content provider inform the broker about their context. In Section 2.3, we selected the MPEG-21 DIA-UED specification to structure and store this context information.

   It is possible to simply wrap the UED-based context information in a SOAP message – illustrated in Listing 3.2 of Section 3.3 – however, this generates overhead due to the verboseness of the plain-text XML notation. Hence, we apply an alternative serialization format before sending the data, as discussed in Chapter 4.

MPEG-B BiM should be used as it provides the best result. As an alternative, the XML-based data can be ZIP compressed, which is the second best alternative serialization technique. The resulting data are added as an attachment to the SOAP message as discussed at the end of Section 3.3.2.

4. The next step is a straightforward request to retrieve information about the available content.

5. The broker embeds its answer (i.e., the overview of the available content) in the SOAP reply. This answer contains at least a (human-readable) title and an identifier for each available video stream. It is necessary that the client knows the structure of the information. For example, it is possible to use a generic container structure, such as MPEG-21 *Digital Item Declaration* [52, 53].

   Different to the architecture described in Section 3.4, we assume that the broker has a local repository containing information on the available content. This repository can be updated by the content provider by sending information about new or updated content. The content provider can use MPEG-7 as content-description tool, as discussed in Section 2.2. Because this information is static and updates thereof can be performed independently from the actual VoD application, we do not consider how to create and update this repository.

6. The client informs the broker about the selected video.

7. In this step, the broker decides on the adaptation rules. As stated in the introduction of Section 3.2, we do not go into detail how to reach an optimal decision; here we use a straightforward mapping algorithm.

   Still, the alternativly serialized UED-based data (received in step 3) must be processed. The serialization-agnostic parser, discussed in Section 4.4, and the MPEG-21 DIA-UED software toolkit, discussion in Section 2.3.2, can be used to simplify this process.

   The resulting adaptation rules are transmitted to the content provider. This can be done by a SOAP message or by an ad hoc solution.

8. The broker sends a SOAP reply containing the URI of the content. This is a reply to the SOAP message of step 6.

9. The client requests the content from the content provider using the URI with the session identifier appended.

10. Finally, the content provider optimizes and streams the selected audio-visual content to the client.

    According to the principles of the UMA concept, it is required to automatically derive different versions from a single content base. This requires content encoded in a scalable way, as discussed in Section 5.3. Unfortunately, no advanced scalable video encoding technique currently has a real-time decoder. As the real-time decoding of the audio-visual stream is a pre-requisite for our application, it is imperative to use a video coding technology that supports this. As such, we selected AVC as the video compression technology – AVC was introduced in Section 5.3.2. This further implies that the content adaptation is limited to temporal scalability for the video stream and enabling or disabling the audio stream.

The technologies selected above can also be used for the steps of the consumption phase depicted in Figure C.3. Indeed, the first step is comparable to the third step of the initialization phase. Furthermore, if MPEG-B BiM is used as an alternative serialization format, the update functionality of BiM can be exploited to further reduce the overhead. The second step of the consumption phase is equal to initialization step 9. Finally, the third step is similar to step 10, however, during the consumption phase the adaptation of the audio-visual stream can not interrupt or restart the stream. Hence, this adaptation must occur on-the-fly.

C.4 Application

In the remainder, we investigate the implementation details and issues to create a VoD application based on the usage scenario and the selected technologies. A technical overview of the architecture and the used technologies of the VoD application is depicted in Figure C.8 at the end of this appendix.

C.4.1 Client

The client is an application with a GUI that allows the user to select and consume a video. In order to demonstrate the on-the-fly and real-time content adaptation, this GUI also allows an end user to emulate changes of the context.

Figure C.4: Client GUI application: select the broker service.

Figure C.4 shows the user interface to choose a broker. Selecting a specific broker invokes the first five steps of the initialization phase. As a result, an overview of the available videos is presented to the user (Figure C.5).

Figure C.5: Client GUI application: select an available video.

After the user selects one of these, the final user interface is shown (Figure C.6). The requested and (dynamically) adapted audio-visual stream is rendered in the center of this GUI. To the left and to the right thereof, several options are presented. These options allow the end user to emulate a change in the context. Clicking on a button results in the creation of an MPEG-21 DIA-UED compliant message that is sent to

the broker. Hence, it emulates step 1a of the consumption phase. The broker re-determines the adaptation rules and sends them to the content provider. The latter re-adapts the audio-visual stream on-the-fly, which becomes visible in the GUI. Table C.1 gives an overview of the mapping of a particular button to the UED-based information that is sent to the broker.

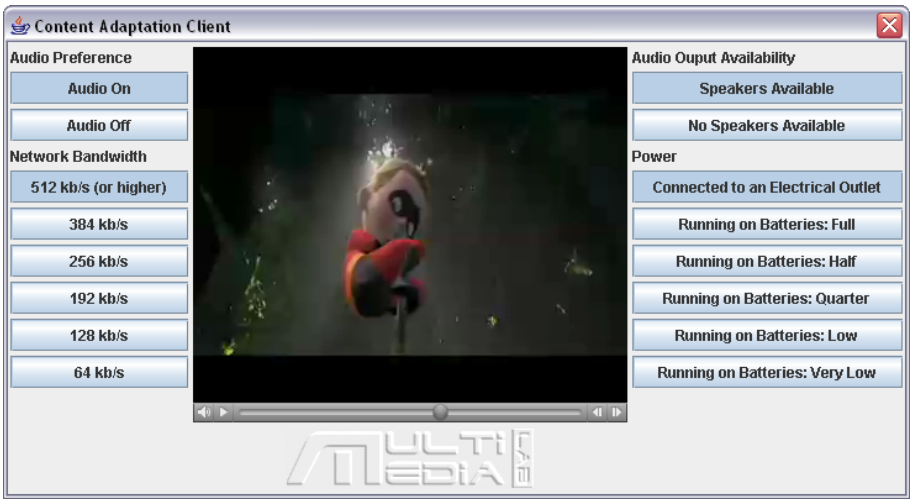


Figure C.6: Client GUI application: playing the selected video.

The client application was developed in the J2SE version 5.0 programming language. The rendering of the audio-visual stream was handled by Apple's QuickTime player, which was embedded in the GUI using the *QuickTime for Java* API [148]. All SOAP handling was performed by the *Java Web Services Developer Pack*[4], in particular, we used version 1.3 of the *SOAP with Attachments API for Java* (SAAJ) library to send the context information. Our client also uses our MPEG-21 DIA-UED software toolkit, as discussed in Section 2.3.2, to simplify the creation of UED-compliant messages. Finally, the client uses BiM as alternative serialization format to reduce the overhead as discussed in Chapter 4 and recommended in previous section. The BiM encoding of the UED-based data is performed by the BiM reference software [98], similar to Use Case 1 in the evaluation of the alternative XML serialization formats in

[4]More information on the Java Web Service Developer Pack is available at `http://java.sun.com/webservices/jwsdp`.

Section 4.5. The client application exploits the update functionality of BiM to further reduce the overhead of the context information.

C.4.2 Network

The network was modeled by the *Click Modular Router*[5]. This router makes it possible to create advanced real-life network topologies in order to emulate existing networks. It comes with various *network elements*, each emulating a specific part of a network, for example a *queue*.

To test our application, we implemented a straightforward set up in the Click router whereby the *PacketBandwidthShaper* element plays the most important role (Figure C.7). This network element – developed by the Department of Information Technology of Ghent University – makes it possible to modify the available bandwidth of the Click router on-the-fly and does this in an intelligent way that closely resembles real bandwidth fluctuations in a network. The router allows us to emulate various other network conditions, such as jitter, error ratio, and packet loss. Nevertheless, we have confined ourselves to support only modifications in the available bandwidth as proof of concept.

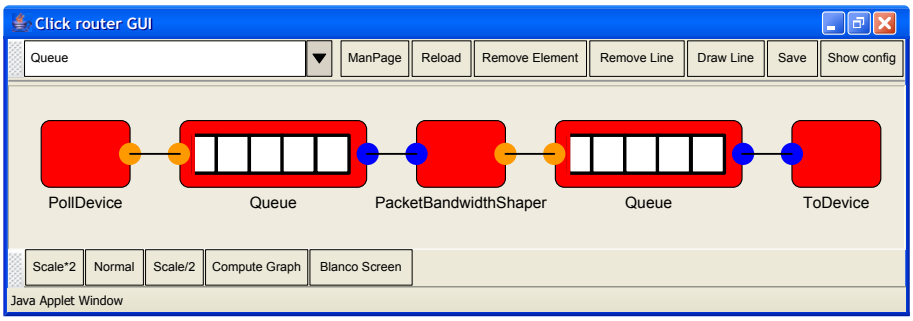


Figure C.7: The implemented network topology using the Click Modular Router.

Additionally, we developed an application that monitors the available bandwidth of the (emulated) network by reading out the value of the PacketBandwidthShaper element in Click. This application notifies the broker whenever a change of the bandwidth occurs. This monitoring tool was developed in the Java programming language and, similar to

[5]More information on the Click Modular Router is available at `http://www.read.cs.ucla.edu/click`.

Table C.1: Client application: mapping rules of the GUI context emulation buttons to a UED-compliant context message.

GUI category	GUI selection	UED name	UED parameter	UED value
Audio Preference	Audio On	audio presentation preference	volume control	1
	Audio Off			0
Audio Output Availability	Speakers Available	audio output capabilities	number of channels	2
	No Speakers Available			0
Network Bandwidth	512 kb/s (or higher)	network capability	available bandwidth	512000
	384 kb/s			384000
	256 kb/s			256000
	192 kb/s			192000
	128 kb/s			128000
	64 kb/s			64000
Power	Connected to an Electrical Outlet	power characteristic	running on batteries	false
	Running on Batteries: ...			true
	... Full	power characteristic	time remaining	3600
	... Half			1800
	... Quarter			600
	... Low			60
	... Very Low			30

the client GUI, all SOAP handling was performed by the SAAJ library. In order to create MPEG-21 DIA-UED compliant information about the network status to the broker, the network also makes use of our UED software toolkit. This results in UED-compliant context information, similar to Listing A.2 in Appendix A.

In order to reduce the overhead, we preferred the MPEG-B BiM technology – as recommended in previous section – using the BiM reference software. Unfortunately, the reference software only works on Microsoft Windows operating systems because it calls functionalities stored in precompiled Windows-based libraries. As Click runs on a Linux-based operating system, it is not feasible to use the BiM reference software. There are currently no alternative BiM encoders available. Hence, we used the second best alternative serialization format, namely ZIP compression.

C.4.3 Broker

The broker exposes four web services that can be invoked. These services handle all incoming requests of the broker (represented by the incoming arrows in Figure C.2 and Figure C.3). The outcome of the web services is sent to the originator of the request, this is represented by the broker's outgoing arrows in the aforementioned figures.

The first three web services are related to steps in the initialization phase. The web service `register` (Table C.2) takes care of the incoming request of step 1 and its reply is step 2. The web service `getOverview` (Table C.3) handles the request of step 4 and the reply of step 5. The web service `getVideo` (Table C.4) handles the incoming request of step 6 and the reply of step 8.

The fourth web service `setUEDContext` (Table C.5) allows clients and networks to send or update the context information (step 3 of the initialization phase and step 1 of the consumption phase). In order to reduce overhead, it is suggested to use an alternative serialization method. Because the broker transfers the received information to our serialization-agnostic parser, the data may be serialized in any of the following formats: plain-text XML, BiM encoded, ZIP compressed, and ASN.1-PER encoded. In addition, these serialization types may be mixed in different SOAP messages.

The Web service functionality of the broker is implemented using similar

techniques as the client and is complemented with Apache's *Tomcat*[6] Web server to provide the Web service infrastructure.

Table C.2: Broker web service: `register`.

Name	`register`
Parameter(s)	None.
Return value	A session identifier.
Description	Registers a client and returns a unique session identification string.
SOAP request body	`<register/>`
SOAP return body	`<result>`*identifier*`</result>`

Table C.3: Broker web service: `getOverview`.

Name	`getOverview`
Parameter(s)	The session identifier.
Return value	MPEG-21 Digital Item.
Description	Retrieves an overview of the available videos.
SOAP request body	`<getOverview sessionID="`*identifier*`"/>`
SOAP return body	`<DIDL><Container id="Movies">[...]`

The broker is also responsible to determine the adaptation rules and must inform the content provider about its decision. Our content adaptation decision engine is a straightforward mapping algorithm whereby we take particular characteristics of the context into account. The mapping is depicted in Table C.6. We take the context characteristics into account that the client sends out after a button on the GUI is pressed (Table C.1) and the context information transmitted by the network monitoring application. If the network bandwidth in the Click router differs from the selected network bandwidth in the GUI of the client, the broker uses the smallest value.

[6]More information on Tomcat is available at `http://tomcat.apache.org`.

Table C.4: Broker web service: `getVideo`.

Name	`getVideo`
Parameter(s)	The session and video identifier.
Return value	The URI.
Description	Retrieves the URI of the given video.
SOAP request body	`<getVideo sessionID="`*identifier*`" videoID="`*video identifier*`"/>`
SOAP return body	`<URI>`*rtsp://URIvideo*`</URI>`

Table C.5: Broker web service: `setUEDContext`.

Name	`setUEDContext`
Parameter(s)	The session identifier (optional) and the UED-compliant context as attachment.
Return value	None.
Description	Sets (or updates) the context information with the UED-compliant information. If no session identifier is given, only the network context information can be set or updated.
SOAP request body	`<setUEDContext sessionID="`*identifier*`"/>`
SOAP return body	Empty.

C.4.4 Content Provider

The main task of the content provider is to stream the requested video optimized according to the adaptation rules and to inform the broker about its status. The latter is useful to make load-balancing between different content providers possible. It also makes load-optimization for different streams on one content provider possible. Nevertheless, we did not add this kind of negotiation to our demonstration application, although similar techniques as for the negotiation of the client and the network with the broker could be used.

From the various available streaming servers, we have selected Apple's *Darwin Streaming Server*[7] because this server provides all required base

[7] More information on Darwin Streaming Server is available at `http://developer.apple.com/opensource/server/streaming`.

Table C.6: Content adaptation decision engine: mapping rules from UED-context information to adaptation rules.

stream	adaptation type	UED name*	UED parameter	UED value
audio	off	audio presentation preference	volume control	0
		audio output capabilities	number of channels	0
	on	audio presentation preference	volume control	> 0
		audio output capabilities	number of channels	> 0
video	all frames	power characteristic	running on batteries	false
		power characteristic	running on batteries + time remaining	true ≥ 3600
		network capability	available bandwidth	≥ 512000
	drop B-frames	power characteristic	running on batteries + time remaining	true ≥ 1800 & < 3600
		network capability	available bandwidth	≥ 384000 & < 512000
	drop B- & 25% P-frames	power characteristic	running on batteries + time remaining	true ≥ 600 & < 1800
		network capability	available bandwidth	≥ 256000 & < 384000
	drop B- & 50% P-frames	power characteristic	running on batteries + time remaining	true ≥ 60 & < 600
		network capability	available bandwidth	≥ 12800 & < 256000
	I-frames only	power characteristic	running on batteries + time remaining	true < 60
		network capability	available bandwidth	≥ 64000 & < 12800
	no video	network capability	available bandwidth	< 64000

* Multiple entries are "or" combined.

functionalities (i.e., it is capable of streaming AVC-encoded bitstreams), flawlessly works together with Apple's QuickTime player that is embedded in the client software, and its C++-based source code is available. The latter is an important advantage as our content adaptation engine can be directly built into the streaming server.

Darwin Streaming Server uses the *Real-Time Transport Protocol* (RTP) [149] to stream video content. This is a one-way protocol solely intended to transmit data. As such, RTP is often complemented with the *Real-Time Streaming Protocol* (RTSP) [150] and the *Real-Time Control Protocol* (RTCP) (also defined in [149]). The former is a kind of "remote control" and supports features as start, pause, stop, rewind, go the next chapter, et cetera. The latter is a feedback mechanism that informs the streaming server about the status of the network, such as jitter and packet loss. It should be noted that RTCP can not be used in our VoD application because the content adaptation decision engine is not located at the streaming server, hence the RTCP feedback can not be taken into account during the content adaptation decision taking process.

Listing C.1 depicts the (most important) code that we have added to the Darwin Streaming Server in order to integrate the content adaptation engine with the streaming server. This code is executed every time an RTP packet is going to be sent.

First, the session identifier is retrieved and stored in the `theSessionID` variable (lines 1 to 4). The content adaptation engine uses this identifier to select the appropriate adaptation rules for this particular client and stream. As such, the adaptation engine can perform different kinds of adaptations for different streams simultaneously.

Next, the payload type of the RTP packet is investigated. In case of audio payload (line 7), our algorithm checks whether or not the audio stream is enabled (line 10). If not, the content of the RTP packet is removed (line 13) and as a result, the Darwin Streaming Server discards this packet.

In case of video payload (line 19), we first perform on lines 21 to 27 a check similar to the audio payload check. The remaining lines exploit the excellent provisions of the Darwin Streaming Server with regard to temporal scalability. The streaming server supports a *quality level* concept for the video stream, which regulates the frame rate. In total, Darwin pre-defines five quality levels (Table C.7). By setting our desired

quality level (determined by the adaptation rules), we can alter the frame rate of the video stream on-the-fly and in real-time.

Listing C.1: Adding real-time content adaptation functionality to the Darwin Streaming Server.

```
1  // Get the Session Identifier of the client & video stream
2  QTSS_RTPPayloadType thePayloadType = (QTSS_RTPPayloadType)
       theLastPacketTrack->Cookie2;
3  char *theSessionID = NULL;
4  theErr = QTSS_GetValueAsString((QTSS_Object)inParams->
       inClientSession, qtssCliSesReqQueryString, 0, &
       theSessionID);
5
6  // check payload: audio?
7  if (thePayloadType == qtssAudioPayloadType)
8  {
9      // check: audio enabled?
10     if (!rdsExtension->getVolumeEnabled(theSessionID))
11     {
12         // audio disabled: discard packet
13         (*theFile)->fPacketStruct.packetData = NULL;
14         continue;
15     }
16 }
17
18 // check payload: video?
19 if (thePayloadType == qtssVideoPayloadType)
20 {
21     // check: video enabled?
22     if (!rdsExtension->getVideoEnabled(theSessionID))
23     {
24         // video disabled: discard packet
25         (*theFile)->fPacketStruct.packetData = NULL;
26         continue;
27     }
28
29     // Retrieve the quality level parameter from the content
           adaptation engine.
30     *theQualityLevel = rdsExtension->getBandwidthParameter(
           theSessionID);
31
32     // Set the quality level
33     (*theFile)->fFile.SetTrackQualityLevel(
           theLastPacketTrack, *theQualityLevel);
34 }
```

Table C.7: Overview of the temporal scalability support in the Darwin Streaming Server.

Keyword	Value	Temporal Scalability
kAllPackets	0	Full frame rate.
kNoBFrames	1	The B-frames are not transmitted.
k75PercentPFrames	2	The B-frames are not transmitted and only the first three quarters of the P-frames are transmitted.
k50PercentPFrames	3	The B-frames are not transmitted and only the first half of the P-frames are transmitted.
kKeyFramesOnly	4	Only the I-frames are transmitted.

The code added to the Darwin Streaming Server contains three method calls to the `rdsExtension` object. Practically, this object processes the adaptation rules received from the broker, maps these rules in a straightforward way to Darwin's quality levels, and keeps an overview of the different client sessions. As these are typical bookkeeping operations, we do not discuss this object in detail.

A technical overview of the architecture and the used technologies is given in Figure C.8.

C.5 Conclusions

In this appendix, we presented a report on the usability of the (novel) technologies discussed and introduced in this thesis by developing a Video-on-Demand application that is constructed according to the UMA principles and capable of handling time-varying metadata. By creating such an application we proved that our novel techniques are (amongst other things) usable to ameliorate the current available VoD solutions.

Based on the results of the research presented in this thesis, we developed a realistic architecture for a VoD application, composed of four components, namely a client, a network, a broker, and a content provider. The interaction between these components is divided in two phases: an initialization phase and a consumption phase. The former ensures the creation of a UMA-compliant application; the latter enables support for

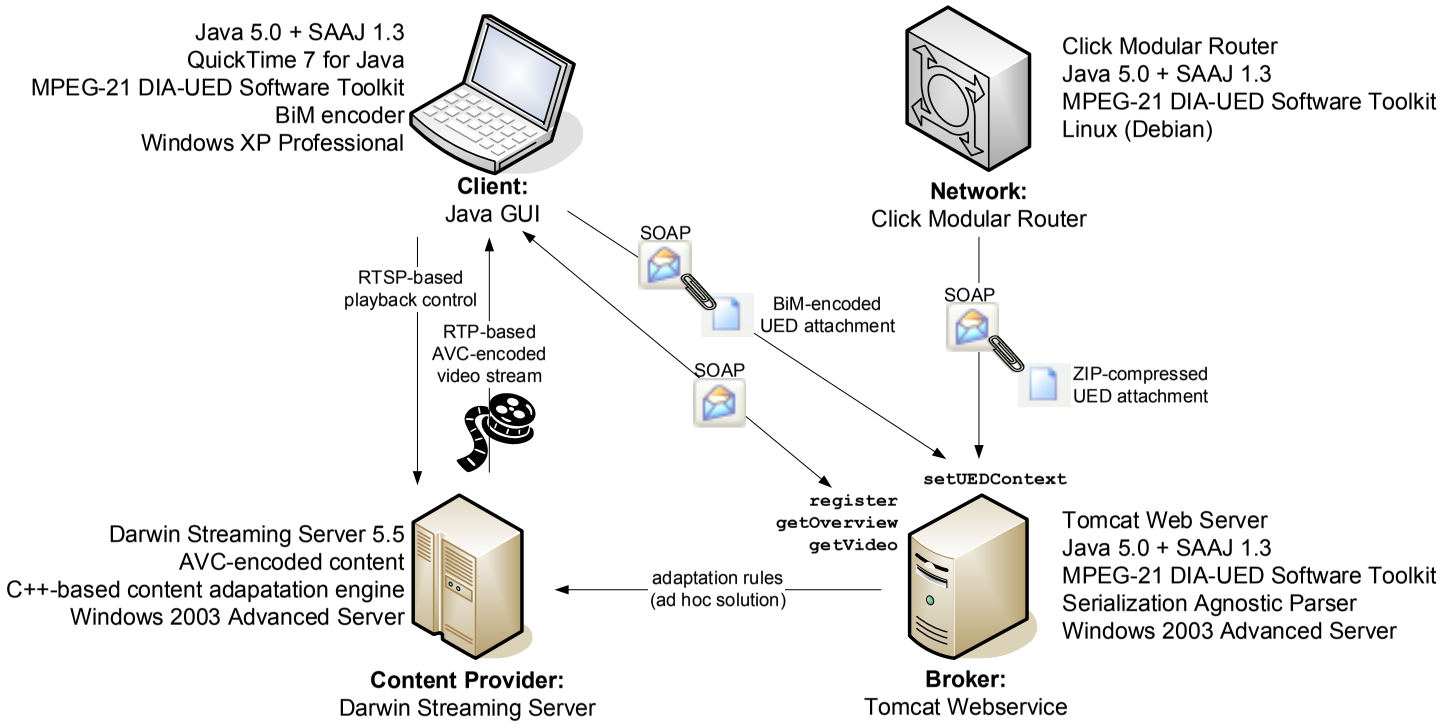


Figure C.8: VoD Application: technical overview.

time-varying metadata.

Next, we selected the technologies that are suitable for the different steps in the two phases. This selection is based on the results of the previous chapters. Finally, we discussed the implementation details of an actual VoD application according to the defined architecture and using the selected technologies. This results in the creation of a VoD application whereby the content is adapted on-the-fly and in real-time to the changing context. A demonstration version of this application is available at the Multimedia Lab research group Website[8].

Although we have achieved our main objective as to prove the applicability of the discussed and novel introduced techniques, some parts could not be realized as desired. The main issue was the lack of real-time decoders for scalable encoded video content; we selected AVC-encoded content as an alternative. As a consequence, our on-the-fly content adaptation is limited to temporal scalability and enabling/disabling of the audio stream. The more advanced scalability schemes – such as the selective degradation of the video using Region-of-Interest enhanced with our fast object tracking technique – could not be demonstrated in this application. Using MPEG-B BiM as the alternative serialization method resulted in a second issue. To BiM encode XML-based data, we use the Windows-based BiM reference software. However, as the network is emulated by a Click Modular Router running on a Linux-based operating system, the reference software could not be used. Unfortunately, there are currently no (commercial or open source) alternative BiM encoders available. Final and third issue, for demonstration purposes we allowed the end user to manually alter the context information. Although this is acceptable and desirable to illustrate the feasibility of our architecture, real-life applications should aggregate the context information automatically. However, no generic solutions for realizing this currently exist.

Notwithstanding the aforementioned issues, we are convinced that our constructed architecture and application illustrates the usability of our results and our novel contributions discussed in this thesis.

[8]The Multimedia Lab research group Website is available at `http://multimedialab.elis.ugent.be`.

Publications

Articles in Journals

1. Robbie De Sutter, Sam Lerouge, Peter De Neve, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Comparison of XML Serializations: Cost Benefit vs. Complexity. *Multimedia Systems*. To appear (DOI : 10.1007/s00530-006-0044-y)

2. Robbie De Sutter, Stijn Notebaert, and Rik Van de Walle. Evaluation of Metadata Standards in the Context of Digital Audio-Visual Libraries. *Lecture Notes in Computer Science*, 4172:220–231, September 2006

3. Robbie De Sutter, Koen De Wolf, Sam Lerouge, and Rik Van de Walle. Lightweight Object Tracking in Compressed Video Streams Demonstrated in Region-of-Interest Coding. *Eurasip Journal on Applied Signal Processing*. To appear

4. Davy De Schrijver, Wesley De Neve, Koen De Wolf, Robbie De Sutter, and Rik Van de Walle. An Optimized MPEG-21 BSDL Framework for the Adaptation of Scalable Bitstreams. *Journal of Visual Communication and Image Representation*. To appear

Papers in Conferences with International Referees (first author)

1. Robbie De Sutter, Hans De Meyer, Bernard De Baets, and Helga Naessens. Fuzzy Similarity Measures and Tree Comparison. In *Proceedings of the Atlantic Symposium on Computational Biol-*

ogy and Genome Information Systems & Technology, pages 87–91, Durham, North Carolina, USA, March 2001

2. Robbie De Sutter, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Adapting Mobile Multimedia Applications to Changing End-User Preferences. In *Proceedings of Euromedia 2002*, pages 180–182, Modena, Italy, April 2002

3. Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Dynamic Adaptation of Multimedia Data for Mobile Applications. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 240–248, Boston, Massachusetts, USA, July 2002

4. Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, and Rik Van de Walle. Dynamic Adaptation of Streaming MPEG-4 Video for Mobile Applications. In *Proceedings of Euromedia 2003*, pages 185–190, Plymouth, United Kingdom, April 2003

5. Robbie De Sutter, Sam Lerouge, Wesley De Neve, Peter Lambert, and Rik Van de Walle. Advanced Mobile Multimedia Applications: using MPEG-21 and Time-Dependent Metadata. In *Proceedings of SPIE/ITCom Multimedia Systems and Applications VI*, volume 5241, pages 147–156, Orlando, Florida, USA, September 2003

6. Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Evaluation of Usage Environment Description Tools. In *Proceedings of the 2004 International Conference on Internet Computing*, pages 66–72, Las Vegas, Nevada, USA, June 2004

7. Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Evaluation of Models for Parsing Binary Encoded XML-based Metadata. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, pages 419–424, Seoul, Korea, November 2004

8. Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Multimedia Metadata Processing: a Format Independent Approach. In *Proceedings of the 9th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 343–348, Grindelwald, Switzerland, February 2005

9. Robbie De Sutter, Sam Lerouge, Davy De Schrijver, and Rik Van de Walle. Enhancing RSS Feeds: Eliminating Overhead through Binary Encoding. In *Proceedings of the IEEE 3rd International Conference on Information Technology and Applications*, pages 520–525, Sydney, Australia, July 2005

10. Robbie De Sutter, Stijn Notebaert, Laurence Hauttekeete, and Rik Van de Walle. IPEA: the Digital Archive Use Case. In *Proceedings of the IS&T Archiving 2006*, pages 182–186, Ottawa, Canada, May 2006

11. Robbie De Sutter and Rik Van de Walle. Saving Bandwidth for RSS Feeds by using ASN.1 in E-learning Applications. In *The 9th IASTED International Conference on Computers and Advanced Technology in Education*, Lima, Peru, October 2006. Accepted for Publication

Papers in Conferences with International Referees (co-author)

1. Jeroen Bekaert, Dimitri Van De Ville, Boris Rogge, Sam Lerouge, Robbie De Sutter, Emiel De Kooning, and Rik Van de Walle. Metadata-based Access to Multimedia Architectural and Historical Archive Collections. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 22–29, Boston, Massachusetts, USA, July 2002

2. Sam Lerouge, Boris Rogge, Robbie De Sutter, Jeroen Bekaert, Dimitri Van De Ville, and Rik Van de Walle. A Generic Mapping Mechanism between Content Description Metadata and User Environments. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 12–21, Boston, Massachusetts, USA, July 2002

3. Boris Rogge, Robbie De Sutter, Jeroen Bekaert, and Rik Van de Walle. An Analysis of Multimedia Formats for Content Description. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 1–11, Boston, Massachusetts, USA, July 2002

4. Jeroen Bekaert, Robbie De Sutter, Rik Van de Walle, and Emiel De Kooning. Metadata-based Access to Complex Digital Objects in Multimedia Archival Collections. In *Proceedings of Euromedia 2003*, pages 10–13, Plymouth, United Kingdom, April 2003

5. Sam Lerouge, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Fully Scalable Video Coding in Multicast Applications. In *Proceedings of SPIE/Electronic Imaging 2004*, volume 5308, pages 555–564, San Jose, California, USA, January 2004

6. Peter Lambert, Lieven Eeckhout, Robbie De Sutter, Koen De Bosschere, and Rik Van de Walle. Low-Level Behavioral Analysis of the JVT/AVC Decoder. In *Proceedings of SPIE/Electronic Imaging 2004*, pages 1371–1382, San Jose, California, USA, January 2004

7. Jo Van Hoecke, Robbie De Sutter, and Paul De Knop. IKGym: Innovation through Co-operation in Quality Management. In *Proceedings of the 12th European Sport Management Congress*, page 294, Ghent, Belgium, September 2004

8. Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. MPEG-21 Session Mobility on Mobile Devices. In *Proceedings of the 2005 International Conference on Internet Computing*, pages 287–293, Las Vegas, Nevada, USA, June 2005

9. Koen De Wolf, Robbie De Sutter, Wesley De Neve, and Rik Van de Walle. Comparison of Prediction Schemes with Motion Information Reuse for Low Complexity Spatial Scalability. In *Proceedings of SPIE/Visual Communications and Image Processing*, volume 5960, pages 1911–1920, Beijing, China, July 2005

10. Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Personalizing Quality Aspects in Scalable Video Coding. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, Amsterdam, The Netherlands, July 2005. Published on CD-ROM

11. Davy De Schrijver, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Lossless Image Coding based on Fractals. In *Proceedings of the 7th IASTED International Conference on Signal and Image Processing*, pages 52–57, Honolulu, Hawaii, USA, August 2005

12. Jo Van Hoecke, Paul De Knop, and Robbie De Sutter. IKGym: A functional Quality System as Management Tool for Voluntary Board Members. In *Proceedings of the 13th Congress of the European Association for Sport Management*, pages 277–278, Newcastle Gateshead, U.K., September 2005

13. Wim Van Lancker, Robbie De Sutter, Davy De Schrijver, and Rik Van de Walle. A Framework for Transformations of XML within the Binary Domain. In *Proceedings of the 10th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 29–34, Innsbruck, Austria, February 2006

14. Jo Van Hoecke, Hugo Schoukens, and Robbie De Sutter. Foot PASS: a Constructive and Distinctive Quality System for Youth Academies of Professional Football Clubs. In *Proceedings of the 14th Congress of the European Association for Sport Management*, Nicosia, Cyprus, September 2006. Accepted for Publication

MPEG Contributions

1. Frederik De Keukelaere, Wesley De Neve, Robbie De Sutter, and Rik Van de Walle. Suggestions Concerning MPEG-21 Digital Item Method Operations and their Implementation. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M9754, Trondheim, Norway, July 2003

2. Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Digital Item Adaptation – Usage Environment Description Tool Parser. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10387, Waikoloa, Hawaii, USA, December 2003

3. Davy De Schrijver, Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. Digital Item Adaptation – Reference Software Tests. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10436, Waikoloa, Hawaii, USA, December 2003

4. Frederik De Keukelaere, Jeroen Bekaert, Patrick Hochstenbach, Robbie De Sutter, Herbert Van de Sompel, and Rik Van de Walle. Issues related to the inclusion of DIP information in DIDs. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10424, Waikoloa, Hawaii, USA, December 2003

5. Jeroen Bekaert, Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. BNB Comments on ISO/IEC 21000-10 CD Part 10: Digital Item Processing. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10621, Munich, Germany, March 2004

6. Christian Timmerer, Stephen Davis, Itaru Kaneko, Spencer Cheng, and Robbie De Sutter. Report of CE on MPEG-21 Binarisations. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10974, Redmond, Washington, USA, July 2004

7. Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Using MPEG-21 Part 16 in Applications. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11325, Palma De Mallorca, Spain, October 2004

8. Christian Timmerer and Robbie De Sutter. CE Report on MPEG-21 Binarization. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11744, Hong Kong, China, January 2005

9. Christian Timmerer, Ingo Kofler, Johannes Liegl, Hermann Hellwagner, Robbie De Sutter, Wim Van Lancker, and Rik Van de Walle. Report of CE on MPEG-21 Binary Format. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11858, Busan, Korea, April 2005

10. Davy De Schrijver, Robbie De Sutter, and Rik Van de Walle. Report on Core Experiment on BSDL extensions. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M12611, Nice, France, October 2005

References

[1] Andrew Perkis, Yousri Abdeljaoued, Charilaos Christopoulos, Touradj Ebrahimi, and Joe F. Chicharo. Universal Multimedia Access from Wired and Wireless Systems. *Circuits, Systems and Signal Processing – Special Issue on Multimedia Communications*, 20(3-4):387–402, May 2001.

[2] Robbie De Sutter, Sam Lerouge, Peter De Neve, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Comparison of XML Serializations: Cost Benefit vs. Complexity. *Multimedia Systems*. To appear (DOI : 10.1007/s00530-006-0044-y).

[3] Robbie De Sutter, Stijn Notebaert, and Rik Van de Walle. Evaluation of Metadata Standards in the Context of Digital Audio-Visual Libraries. *Lecture Notes in Computer Science*, 4172:220–231, September 2006.

[4] Robbie De Sutter, Koen De Wolf, Sam Lerouge, and Rik Van de Walle. Lightweight Object Tracking in Compressed Video Streams Demonstrated in Region-of-Interest Coding. *Eurasip Journal on Applied Signal Processing*. To appear.

[5] Davy De Schrijver, Wesley De Neve, Koen De Wolf, Robbie De Sutter, and Rik Van de Walle. An Optimized MPEG-21 BSDL Framework for the Adaptation of Scalable Bitstreams. *Journal of Visual Communication and Image Representation*. To appear.

[6] Robbie De Sutter, Hans De Meyer, Bernard De Baets, and Helga Naessens. Fuzzy Similarity Measures and Tree Comparison. In *Proceedings of the Atlantic Symposium on Computational Biology and Genome Information Systems & Technology*, pages 87–91, Durham, North Carolina, USA, March 2001.

[7] Robbie De Sutter, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Adapting Mobile Multimedia Applications to Changing End-User Preferences. In *Proceedings of Euromedia 2002*, pages 180–182, Modena, Italy, April 2002.

[8] Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Dynamic Adaptation of Multimedia Data for Mobile Applications. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 240–248, Boston, Massachusetts, USA, July 2002.

[9] Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, and Rik Van de Walle. Dynamic Adaptation of Streaming MPEG-4 Video for Mobile Applications. In *Proceedings of Euromedia 2003*, pages 185–190, Plymouth, United Kingdom, April 2003.

[10] Robbie De Sutter, Sam Lerouge, Wesley De Neve, Peter Lambert, and Rik Van de Walle. Advanced Mobile Multimedia Applications: using MPEG-21 and Time-Dependent Metadata. In *Proceedings of SPIE/ITCom Multimedia Systems and Applications VI*, volume 5241, pages 147–156, Orlando, Florida, USA, September 2003.

[11] Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Evaluation of Usage Environment Description Tools. In *Proceedings of the 2004 International Conference on Internet Computing*, pages 66–72, Las Vegas, Nevada, USA, June 2004.

[12] Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Evaluation of Models for Parsing Binary Encoded XML-based Metadata. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, pages 419–424, Seoul, Korea, November 2004.

[13] Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Multimedia Metadata Processing: a Format Independent Approach. In *Proceedings of the 9th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 343–348, Grindelwald, Switzerland, February 2005.

[14] Robbie De Sutter, Sam Lerouge, Davy De Schrijver, and Rik Van de Walle. Enhancing RSS Feeds: Eliminating Overhead

through Binary Encoding. In *Proceedings of the IEEE 3rd International Conference on Information Technology and Applications*, pages 520–525, Sydney, Australia, July 2005.

[15] Robbie De Sutter, Stijn Notebaert, Laurence Hauttekeete, and Rik Van de Walle. IPEA: the Digital Archive Use Case. In *Proceedings of the IS&T Archiving 2006*, pages 182–186, Ottawa, Canada, May 2006.

[16] Robbie De Sutter and Rik Van de Walle. Saving Bandwidth for RSS Feeds by using ASN.1 in E-learning Applications. In *The 9th IASTED International Conference on Computers and Advanced Technology in Education*, Lima, Peru, October 2006. Accepted for Publication.

[17] Jeroen Bekaert, Dimitri Van De Ville, Boris Rogge, Sam Lerouge, Robbie De Sutter, Emiel De Kooning, and Rik Van de Walle. Metadata-based Access to Multimedia Architectural and Historical Archive Collections. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 22–29, Boston, Massachusetts, USA, July 2002.

[18] Sam Lerouge, Boris Rogge, Robbie De Sutter, Jeroen Bekaert, Dimitri Van De Ville, and Rik Van de Walle. A Generic Mapping Mechanism between Content Description Metadata and User Environments. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 12–21, Boston, Massachusetts, USA, July 2002.

[19] Boris Rogge, Robbie De Sutter, Jeroen Bekaert, and Rik Van de Walle. An Analysis of Multimedia Formats for Content Description. In *Proceedings of SPIE/ITCom Internet Multimedia Management Systems III*, volume 4862, pages 1–11, Boston, Massachusetts, USA, July 2002.

[20] Jeroen Bekaert, Robbie De Sutter, Rik Van de Walle, and Emiel De Kooning. Metadata-based Access to Complex Digital Objects in Multimedia Archival Collections. In *Proceedings of Euromedia 2003*, pages 10–13, Plymouth, United Kingdom, April 2003.

[21] Sam Lerouge, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Fully Scalable Video Coding in Multicast Applications. In

Proceedings of SPIE/Electronic Imaging 2004, volume 5308, pages 555–564, San Jose, California, USA, January 2004.

[22] Peter Lambert, Lieven Eeckhout, Robbie De Sutter, Koen De Bosschere, and Rik Van de Walle. Low-Level Behavioral Analysis of the JVT/AVC Decoder. In *Proceedings of SPIE/Electronic Imaging 2004*, pages 1371–1382, San Jose, California, USA, January 2004.

[23] Jo Van Hoecke, Robbie De Sutter, and Paul De Knop. IKGym: Innovation through Co-operation in Quality Management. In *Proceedings of the 12th European Sport Management Congress*, page 294, Ghent, Belgium, September 2004.

[24] Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. MPEG-21 Session Mobility on Mobile Devices. In *Proceedings of the 2005 International Conference on Internet Computing*, pages 287–293, Las Vegas, Nevada, USA, June 2005.

[25] Koen De Wolf, Robbie De Sutter, Wesley De Neve, and Rik Van de Walle. Comparison of Prediction Schemes with Motion Information Reuse for Low Complexity Spatial Scalability. In *Proceedings of SPIE/Visual Communications and Image Processing*, volume 5960, pages 1911–1920, Beijing, China, July 2005.

[26] Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Personalizing Quality Aspects in Scalable Video Coding. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, Amsterdam, The Netherlands, July 2005. Published on CD-ROM.

[27] Davy De Schrijver, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Lossless Image Coding based on Fractals. In *Proceedings of the 7th IASTED International Conference on Signal and Image Processing*, pages 52–57, Honolulu, Hawaii, USA, August 2005.

[28] Jo Van Hoecke, Paul De Knop, and Robbie De Sutter. IKGym: A functional Quality System as Management Tool for Voluntary Board Members. In *Proceedings of the 13th Congress of the European Association for Sport Management*, pages 277–278, Newcastle Gateshead, U.K., September 2005.

[29] Wim Van Lancker, Robbie De Sutter, Davy De Schrijver, and Rik Van de Walle. A Framework for Transformations of XML

within the Binary Domain. In *Proceedings of the 10th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 29–34, Innsbruck, Austria, February 2006.

[30] Jo Van Hoecke, Hugo Schoukens, and Robbie De Sutter. Foot PASS: a Constructive and Distinctive Quality System for Youth Academies of Professional Football Clubs. In *Proceedings of the 14th Congress of the European Association for Sport Management*, Nicosia, Cyprus, September 2006. Accepted for Publication.

[31] Frederik De Keukelaere, Wesley De Neve, Robbie De Sutter, and Rik Van de Walle. Suggestions Concerning MPEG-21 Digital Item Method Operations and their Implementation. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M9754, Trondheim, Norway, July 2003.

[32] Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Digital Item Adaptation – Usage Environment Description Tool Parser. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10387, Waikoloa, Hawaii, USA, December 2003.

[33] Davy De Schrijver, Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. Digital Item Adaptation – Reference Software Tests. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10436, Waikoloa, Hawaii, USA, December 2003.

[34] Frederik De Keukelaere, Jeroen Bekaert, Patrick Hochstenbach, Robbie De Sutter, Herbert Van de Sompel, and Rik Van de Walle. Issues related to the inclusion of DIP information in DIDs. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10424, Waikoloa, Hawaii, USA, December 2003.

[35] Jeroen Bekaert, Frederik De Keukelaere, Robbie De Sutter, and Rik Van de Walle. BNB Comments on ISO/IEC 21000-10 CD Part 10: Digital Item Processing. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10621, Munich, Germany, March 2004.

[36] Christian Timmerer, Stephen Davis, Itaru Kaneko, Spencer Cheng, and Robbie De Sutter. Report of CE on MPEG-21 Binarisations. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M10974, Redmond, Washington, USA, July 2004.

[37] Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Using MPEG-21 Part 16 in Applications. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11325, Palma De Mallorca, Spain, October 2004.

[38] Christian Timmerer and Robbie De Sutter. CE Report on MPEG-21 Binarization. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11744, Hong Kong, China, January 2005.

[39] Christian Timmerer, Ingo Kofler, Johannes Liegl, Hermann Hellwagner, Robbie De Sutter, Wim Van Lancker, and Rik Van de Walle. Report of CE on MPEG-21 Binary Format. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M11858, Busan, Korea, April 2005.

[40] Davy De Schrijver, Robbie De Sutter, and Rik Van de Walle. Report on Core Experiment on BSDL extensions. MPEG Contribution ISO/IEC JTC1/SC29/WG11 M12611, Nice, France, October 2005.

[41] Jerome McDonough, Merrilee Proffitt, and MacKenzie Smith. Structural, Technical, and Administrative Metadata Standards. A Discussion Document. Technical Report, Digital Library Federation, December 2000. Available at `http://www.diglib.org/standards/stamdframe.htm`.

[42] European Broadcasting Union. Metadata Exchange Scheme, v1.0. Technical Report No. 290, April 2002. Available at `http://www.ebu.ch/trev_290-hopper.pdf`.

[43] European Broadcasting Union. ESCORT: EBU System of Classification of RTV Programmes. Technical Report, October 1995. Available at `http://www.ebu.ch/en/technical/metadata/specifications`.

[44] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). Technical Report World Wide Web Consortium (W3C) (Recommendation), February 2004. Available at `http://www.w3.org/TR/2004/REC-xml-20040204`.

[45] David C. Fallside and Priscilla Walmsley. XML Schema Part 0: Primer Second Edition. Technical Report World Wide Web Con-

sortium (W3C) (Recommendation), February 2004. Available at http://www.w3.org/TR/2004/REC-xmlschema-0-20041028.

[46] James Clark. XSL Transformations (XSLT). Technical Report World Wide Web Consortium (W3C) (Recommendation), November 1999. Available at http://www.w3.org/TR/1999/REC-xslt-19991116.

[47] Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, Version 1.1: Reference Description. Technical Report, 2004. Available at http://www.dublincore.org/documents/dces.

[48] Frank Manola, Eric Miller, and Brian McBride. Resource Description Framework (RDF) Primer. Technical Report World Wide Web Consortium (W3C) (Recommendation), February 2004. Available at http://www.w3.org/TR/2004/REC-rdf-primer-20040210.

[49] B. S. Manjunath, Philippe Salembier, and Thomas Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Language*. John Wiley & Sons, June 2002.

[50] José M. Martínez, Rob Koenen, and Fernando Pereira. MPEG-7: The Generic Multimedia Content Description Standard, Part 1. *IEEE Multimedia*, 9(2):78–87, April 2002.

[51] José M. Martínez. MPEG-7: Overview of MPEG-7 Description Tools, Part 2. *IEEE Multimedia*, 9(3):83–93, July 2002.

[52] Ian Burnett, Rik Van de Walle, Keith Hill, Jan Bormans, and Fernando Pereira. MPEG-21: Goals and Achievements. *IEEE Multimedia*, 10(4):60–70, October 2003.

[53] Ian Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen, editors. *The MPEG-21 Book*. John Wiley & Sons, March 2006.

[54] Philippe Salembier and John R. Smith. MPEG-7 Multimedia Description Schemes. *IEEE Transactions on Circuits, Systems and Video Technology*, 11(6):748–759, 2001.

[55] European Broadcasting Union. P/Meta Metadata Exchange Scheme v1.1. Technical Report No. 3295, June 2005. Available at http://www.ebu.ch/en/technical/metadata/specifications/notes_on_tech3295.php.

[56] European Broadcasting Union. Metadata Exchange Standards. Technical Report No. 284, September 2000. Available at `http://www.ebu.ch/en/technical/trev/trev_284-hopper.pdf`.

[57] Tim Berners-Lee, Roy Fielding, and Frystyk Henrik. Hypertext Transfer Protocol - HTTP/1.0. Internet Engineering Task Force Request for Comment 1945, May 1996.

[58] Roy Fielding, James Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1. Internet Engineering Task Force Request for Comment 2616, June 1999.

[59] Graham Klyne, Franklin Reynolds, Chris Woodrow, Hidetaka Ohto, Johan Hjelm, Mark H. Butler, and Luu Tran. Composite Capability / Preference Profile (CC/PP): Structure and Vocabularies. Technical Report World Wide Web Consortium (W3C) (Recommendation), January 2004. Available at `http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/`.

[60] Open Mobile Alliance. UAProf User Agent Profiling Specification. Report No. WAP-248-UAPROF-20011020-a, October 2001. Available at `http://www.openmobilealliance.org`.

[61] ISO/IEC. Information Technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation. ISO/IEC International Standard 21000-7:2004, October 2004.

[62] Anthony Vetro and Christian Timmerer. Digital Item Adaptation: Overview of Standardization and Research Activities. *IEEE Transactions on Multimedia – Special Issue on MPEG-21*, 7(3):435–445, June 2005.

[63] Kim Topley. *J2ME in a nutshell*. O'Reilly, 2002.

[64] V. Sreenivasulu. The Role of a Digital Librarian in the Management of Digital Information Systems (DIS). *Aslib Proceeding*, 18(1):12–20, 2000.

[65] Digital Library Federation. METS: Metadata Encoding and Transmission Standard. Technical Report, November 2005. Available at `http://www.loc.gov/standards/mets`.

[66] Digital Library Federation. The Making of America II. Technical Report, November 2005. Available at `http://sunsite.berkeley.edu/MOA2`.

[67] Library of Congress. *Understanding MARC Authority Records.* Cataloging Distribution Service, June 2003.

[68] The Society of American Archivists. *Encoded Archival Description: Tag Library.* Society of American Archivists, June 2002.

[69] International Council on Archives. *ISAD(G): General International Standard Archival Description, Second edition.* September 1999.

[70] Jeroen Bekaert, Dimitri Van De Ville, Boris Rogge, Iwan Strauven, Emiel De Kooning, and Rik Van de Walle. Metadata-based Access to Multimedia Architectural and Historical Archive Collections: a Review. *Aslib Proceeding*, 54(6):362–371, December 2002.

[71] Shien-Chiang Yu, Hsueh-hua Chen, and Huai-wen Chang. Building an Open Archive Union Catalog for Digital Archives. *The Electronic Library*, 23(4):410–418, August 2005.

[72] Herbert Van de Sompel and Carl Lagoze. The Santa Fe Convention of the Open Archives Initiative. *D-Lib Magazine*, 6(2), February 2000.

[73] Carl Lagoze and Herbert Van de Sompel. The Making of the Open Archives Initiative Protocol for Metadata Harvesting. *Library Hi Tech*, 21(2):118–128, 2003.

[74] Sam Lerouge, Peter Lambert, and Rik Van de Walle. Multi-criteria Optimization for Scalable Bitstreams. In *the 8th International Workshop on Visual Content Processing and Representation*, volume 2849 of *Lecture Notes in Computer Science*, September 2003.

[75] Sam Lerouge. *Personalizing Quality Aspects for Video Communication in Constrained Heterogeneous Environments.* PhD thesis, Ghent University, November 2005.

[76] Bernd Girod, Anne Aaron, Shantanu Rane, and David Rebollo-Monedero. Distributed Video Coding. *Proceedings of the IEEE, Special Issue on Advances in Video Coding and Delivery*, 93(1):71–83, January 2005.

[77] Nilo Mitra. SOAP Version 1.2 Part 0: Primer. Technical Report World Wide Web Consortium (W3C) (Recommendation), June 2003. Available at http://www.w3.org/TR/2003/REC-soap12-part0-20030624.

[78] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. Technical Report World Wide Web Consortium (W3C) (Recommendation), June 2003. Available at http://www.w3.org/TR/2003/REC-soap12-part1-20030624.

[79] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 2: Adjuncts. Technical Report World Wide Web Consortium (W3C) (Recommendation), June 2003. Available at http://www.w3.org/TR/2003/REC-soap12-part2-20030624.

[80] James Clark and Steve DeRose. XML XPath Language. Technical Report World Wide Web Consortium (W3C) (Recommendation), November 1999. Available at http://www.w3.org/TR/1999/REC-xpath-19991116.

[81] Bert Bos. The XML Data Model. Technical Report World Wide Web Consortium (W3C) (Essay), April 1997. Available at http://www.w3.org/XML/datamodel.html.

[82] Arnaud Le Hors, Philippe Le Hégaret, Lauren Wood, Gavin Nicol, Jonathan Robie, Mike Champion, and Steve Byrne. Document Object Model (DOM) Level 3 Core Specification. Technical Report World Wide Web Consortium (W3C) (Recommendation), April 2004. Available at http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407.

[83] Peter Deutsch. DEFLATE Compressed Data Format Specification version 1.3. Internet Engineering Task Force Request for Comment 1951, May 1996.

[84] David A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the I.R.E.*, pages 1098–1102, September 1952.

[85] Jacob Ziv and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[86] Jacob Ziv and Abraham Lempel. Compression of Individual Sequences via Variable Rate Coding. *IEEE Transactions on Information Theory*, 24(5):530–535, 1978.

[87] Peter Deutsch and Jean-Loup Gailly. ZLIB Compressed Data Format Specification version 3.3. Internet Engineering Task Force Request for Comment 1950, May 1996.

[88] Mark Nelson and Jean-Loup Gailly. *The Data Compression Book – second edition*. M&T Books, April 1995.

[89] Khalid Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers, January 1996.

[90] ITU-T and ISO/IEC. Information Technology – Abstract Syntax Notation One (ASN.1) Specification of Basic Notation. Report No. ITU-T Rec. X.680 (2002), ISO/IEC 8824-1:2002, 2002. Available at `http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf`.

[91] ITU-T and ISO/IEC. Information Technology – ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). Report No. ITU-T Rec. X.690 (2002), ISO/IEC 8825-1:2002, 2002. Available at `http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf`.

[92] ITU-T and ISO/IEC. Information Technology – ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER). Report No. ITU-T Rec. X.691 (2002), ISO/IEC 8825-2:2002, 2002. Available at `http://www.itu.int/ITU-T/studygroups/com17/languages/X.691-0207.pdf`.

[93] ITU-T and ISO/IEC. Information Technology – ASN.1 Encoding Rules: XML Encoding Rules (XER). Report No. ITU-T Rec. X.693 (2001), ISO/IEC 8825-4:2001, 2002. Available at `http://www.itu.int/ITU-T/studygroups/com17/languages/X.693-0112.pdf`.

[94] ITU-T and ISO/IEC. Information Technology – ASN.1 Encoding Rules: Mapping W3C XML Schema Definitions into ASN.1. Report No. ITU-T Rec. X.694 (2004), ISO/IEC 8825-5:2004, 2004. Available at `http://www.itu.int/ITU-T/studygroups/com17/languages/X694.pdf`.

[95] Ulrich Niedermeier, Jörg Heuer, Andreas Hutter, Walter Stechele, and Andre Kaup. An MPEG-7 Tool for Compression and Streaming of XML Data. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1, pages 521–524, Lausanne, Switzerland, August 2002.

[96] Jane Hunter. An Overview of the MPEG-7 Description Definition Language (DDL). *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):765–772, June 2001.

[97] ISO/IEC. Information Technology – MPEG Systems Technologies – Part 1: Binary MPEG Format for XML. ISO/IEC International Standard 23001-1:2006, March 2006.

[98] ISO/IEC. Information Technology – Multimedia Content Description Interface – Part 6: Reference Software. ISO/IEC International Standard 15938-6:2003, June 2003.

[99] Mike Cokus and Santiago Pericas-Geertsen. XML Binary Characterization Use Cases. Technical Report World Wide Web Consortium (W3C) (Working Group Note), March 2005. Available at `http://www.w3.org/TR/2005/NOTE-xbc-use-cases-20050331`.

[100] Stephen D. Williams and Peter Haggar. XML Binary Characterization Measurement Methodologies. Technical Report World Wide Web Consortium (W3C) (Working Group Note), March 2005. Available at `http://www.w3.org/TR/2005/NOTE-xbc-measurement-20050331`.

[101] Martin Gudgin, Noah Mendelsohn, Mark Nottingham, and Hervé Ruellan. XML-binary Optimized Packaging. Technical Report World Wide Web Consortium (W3C) (Recommendation), January 2005. Available at `http://www.w3.org/TR/2005/REC-xop10-20050125`.

[102] Anish Karmarkar, Martin Gudgin, and Yves Lafon. Resource Representation SOAP Header Block. Technical Report World Wide Web Consortium (W3C) (Recommendation), January 2005. Available at `http://www.w3.org/TR/2005/REC-soap12-rep-20050125`.

[103] Bruce Martin and Bashar Jano. WAP Binary XML Content Format. Technical Report World Wide Web Consortium (W3C)

(Note), June 1999. Available at `http://www.w3.org/1999/06/NOTE-wbxml-19990624`.

[104] Paul Sandoz, Alessando Triglia, and Santiago Pericas-Geertsen. Fast Infoset. Sun Developer Network Technical Article, June 2004. Available at `http://java.sun.com/developer/technicalArticles/xml/fastinfoset`.

[105] Paul Sandoz, Santiago Pericas-Geertsen, Kohuske Kawaguchi, Marc Hadley, and Eduardo Pelegri-Llopart. Fast Web Services. Sun Developer Network Technical Article, August 2003. Available at `http://java.sun.com/developer/technicalArticles/xml/fastWS`.

[106] Hartmut Liefke and Dan Suciu. XMill: an Efficient Compressor for XML Data. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153–164, May 2000.

[107] James Cheney. Compressing XML with Multiplexed Hierarchical PPM Models. In *Proceedings of IEEE Data Compression Conference*, pages 163–172, October 2001.

[108] Christian Timmerer, Ingo Kofler, Johannes Liegl, and Hermann Hellwagner. An Evaluation of Existing Metadata Compression And Encoding Technologies for MPEG-21 Applications. In *Proceedings of the 7th IEEE International Symposium on Multimedia*, pages 534–539, December 2005.

[109] Anthony Vetro, Charilaos Christopoulos, and Huifang Sun. Video Transcoding Architectures and Techniques: an Overview. *IEEE Signal Processing Magazine*, 20(2):32–36, March 2003.

[110] ISO/IEC. Information Technology – JPEG 2000 Image Coding System – Part 3: Motion JPEG 2000. ISO/IEC International Standard 15444-3:2002, November 2004.

[111] Iain Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. John Wiley & Sons, May 2002.

[112] Weiping Li. Overview of Fine Granularity Scalability in MPEG-4 Video Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):301–317, March 2001.

[113] Fernando Pereira and Touradj Ebrahimi, editors. *The MPEG-4 Book*. Prentice Hall, July 2002.

[114] Mihaele van der Schaar and Yun-Ting Lin. Content-based Selective Enhancement for Streaming Video. In *IEEE International Conference on Image Processing*, volume 2, pages 977–980, October 2001.

[115] Hayder Radha, Mihaela van der Schaar, and Shirish Karande. Scalable Video Transcaling for the Wireless Internet. *Eurasip Journal on Applied Signal Processing*, 2004(2):265–279, February 2004.

[116] Atul Puri and Tsuhan Chen, editors. *Multimedia Systems, Standards, and Networks*. Marcel Dekker Inc., March 2000.

[117] Marek Domanski, Lukasz Blaszak, and Slawomir Mackowiak. AVC Video Coders with Spatial and Temporal Scalability. In *Proceedings of Picture Coding Symposium*, pages 41–46, April 2003.

[118] Kemal Ugur, Giorgos Louizis, Panos Nasiopoulos, and Rabab Ward. Extremely Fast Selective Enhancement Method for Fine Granular Scalable Enabled H.264 Video. In *IEEE Canadian Conference on Electrical and Computer Engineering*, volume 3, pages 1103–1106, May 2003.

[119] Kemal Ugur and Panos Nasiopoulos. Combining Bitstream Switching and FGS for H.264 Scalable Video Transmission Over Varying Bandwidth Networks. In *IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, volume 2, pages 972–975, August 2003.

[120] Joao Ascenso and Fernando Pereira. Drift Reduction for a H.264-AVC Fine Grain Scalability with Motion Compensation Architecture. In *IEEE International Conference on Image Processing*, volume 4, pages 2259–2262, October 2004.

[121] Ke Shen and Edward J. Delp. Wavelet Based Rate Scalable Video Compression. *IEEE Tansactions on Circuits and Systems for Video Technology*, 9(1):109–122, February 1999.

[122] Hongyang Chao and Ming Wei. Rate Scalable Video Compression based on Flexible Block Wavelet Coding Technique. In *IEEE*

4th Workshop on Multimedia Signal Processing, pages 415–420, October 2001.

[123] Osama K. Al-Shaykh, Eugene Miloslavsky, Toshio Nomura, Ralph Neff, and Avideh Zakhor. Video Compression Using Matching Pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):123–143, February 1999.

[124] Fan Ling, Weiping Li, and Hongqiao Sun. Bitplane Coding of DCT Coefficients for Image and Video Compression. In *Proceedings of SPIE Visual Communications and Image Processing*, pages 500–508, January 1999.

[125] Peter Lambert, Wesley De Neve, Philippe De Neve, Ingrid Moerman, Piet Demeester, and Rik Van de Walle. Rate-Distortion Performance of H.264/AVC Compared to State-of-the-Art Video Codecs. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(1):134–140, January 2006.

[126] Thomas Wiegand, Gary Sullivan, Julien Reichel, Heiko Schwarz, and Mathias Wien. Text of ISO/IEC 14496-10:2006/PDAM3 Scalable Video Coding. MPEG Output Document ISO/IEC JTC1/SC29/WG11 N7795, Bangkok, Thailand, January 2006.

[127] Julien Reichel, Heiko Schwarz, and Mathias Wien. Joint Scalable Video Model JSVM-6. MPEG Output Document ISO/IEC JTC1/SC29/WG11 N8015, Montreux, Switzerland, April 2006.

[128] Koen De Wolf, Davy De Schrijver, Wesley De Neve, and Rik Van de Walle. Adaptive Residual Interpolation: a Tool for Efficient Spatial Scalable Video Coding. In *The International Conference on Image Processing, Computer Vision, & Pattern Recognition*, volume 1, pages 131–137, June 2006.

[129] Seung-Jong Choi and John W. Woods. Motion-Compensated 3-D Subband Coding of Video. *IEEE Transactions on Image Processing*, 8(2):155–167, February 1999.

[130] Jens-Rainer Ohm, Mihaela van der Schaar, and John W. Woods. Interframe Wavelet Coding – Motion Picture Representation for Universal Scalability. *Elsevier Journal on Signal Processing: Image Communication*, 19:877–908, 2004.

[131] Jens-Rainer Ohm. Advances in Scalable Video Coding. In *Proceedings of The IEEE*, volume 93, pages 42–56, January 2005.

[132] Fabio Cavalli, Rita Cucchiara, Massimo Piccardi, and Prati Andrea. Performance Analysis of MPEG-4 Decoder and Encoder. In *4th EURASIP - IEEE International Symposium on Video, Image Processing and Multimedia Communications*, pages 227–231, June 2002.

[133] Olli Lehtoranata and Timo D. Hämäläinen. Complexity Analysis of Spatially Scalable MPEG-4 Encoder. In *IEEE International Symposium on System-on-Chip*, pages 57–60, November 2003.

[134] Barry G. Haskell, Atul Puri, and Arun N. Netravali, editors. *Digital Video: an Introduction to MPEG-2*. Chapman and Hall, December 1996.

[135] Stamatia Dasiopoulou, Vasileios Mezaris, Ioannis Kompatsiaris, Vasileios-Kyriakos Papastathis, and Michael G. Strintzis. Knowledge-Assisted Semantic Video Object Detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1210–1224, October 2005.

[136] Hualu Wang and Shih-Fu Chang. A Highly Efficient System for Automatic Face Region Detection in MPEG Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):615–628, August 1997.

[137] Christoph Bregler. Learning and Recognizing Human Dynamics in Video Sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 568–574, June 1997.

[138] Andrea Cavallaro, Olivier Steiger, and Touradj Ebrahimi. Semantic Video Analysis for Adaptive Content Delivery and Automatic Description. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1200–1209, October 2005.

[139] Alain Lipton, Hironobu Fujiyoshi, and Raju Patil. Moving Target Classification and Tracking from Real-Time Video. In *Proceedings of the fourth IEEE Workshop on Applications of Computer Vision*, pages 8–14, October 1998.

[140] Shao-Yi Chien, Yu-Wen Huang, and Liang-Gee Chen. Predictive Watershed: A Fast Watershed Algorithm for Video Segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(5):453–461, May 2003.

[141] Orachat Sukmarg and Kamisetty R. Rao. Fast Object Detection and Segmentation in MPEG Compressed Domain. In *TENCON 2000 Proceedings*, pages 364–368, September 2000.

[142] Vasileios Mezaris, Ionnis Kompatsiaris, Nikolaos V. Boulgouris, and Michael G. Strintzis. Real-Time Compressed-Domain Spatiotemporal Segmentation and Ontologies for Video Indexing and Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):606–621, May 2004.

[143] Michael Isard and Andrew Blake. Contour Tracking by Stochastic Propagation of Conditional Density. In *Proceedings of the European Conference on Computer Vision*, pages 343–356, April 1996.

[144] Wen-Nung Lie and Ruey-Lung Chen. Tracking Moving Objects in MPEG-Compressed Videos. In *IEEE International Conference on Multimedia and Expo*, pages 1172–1175, August 2001.

[145] Radhakrishna Achanta, Mohan Kankanhalli, and Phillippe Mulhem. Compressed Domain Object Tracking for Automatic Indexing of Objects in MPEG Home Video. In *IEEE International Conference on Multimedia and Expo*, pages 61–64, August 2002.

[146] Sung-Mo Park and Joonwhoan Lee. Object Tracking in MPEG Compressed Video using Mean-Shift Algorithm. In *Proceedings of Joint Conference of the 4th International Conference on Information, Communications and Signal Processing*, volume 2, pages 748–752, December 2003.

[147] Lorenzo Favalli, Alessandro Mecocci, and Fulvio Moschetti. Object Tracking for Retrieval Applications in MPEG-2. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):427–432, April 2000.

[148] Tom Maremaa and William Stewart. *QuickTime for Java: A Developer Reference*. Morgan Kaufmann, August 1999.

[149] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications.

Internet Engineering Task Force Request for Comment 3550, July 2003.

[150] Henning Schulzrinne, Anup Rao, and Robert Lanphier. Real Time Streaming Protocol (RTSP). Internet Engineering Task Force Request for Comment 2326, August 1998.